Deciding security properties for cryptographic protocols. Application to key cycles

HUBERT COMON-LUNDH
ENS CACHAN & Research Center for Information Security, AIST, Tokyo
and
VÉRONIQUE CORTIER
LORIA, CNRS & Université Henri Poincaré & INRIA Project CASSIS
and
EUGEN ZĂLINESCU
MSR-INRIA Joint Centre, Orsay

There is a large amount of work dedicated to the formal verification of security protocols. In this paper, we revisit and extend the NP-complete decision procedure for a bounded number of sessions. We use a, now standard, deducibility constraint formalism for modeling security protocols. Our first contribution is to give a simple set of constraint simplification rules, that allows to reduce any deducibility constraint system to a set of *solved forms*, representing all solutions (within the bound on sessions).

As a consequence, we prove that deciding the existence of key cycles is NP-complete for a bounded number of sessions. The problem of key-cycles has been put forward by recent works relating computational and symbolic models. The so-called *soundness* of the symbolic model requires indeed that no key cycle (e.g., $\mathrm{enc}(k, k)$) ever occurs in the execution of the protocol. Otherwise, stronger security assumptions (such as KDM-security) are required.

We show that our decision procedure can also be applied to prove again the decidability of authentication-like properties and the decidability of a significant fragment of protocols with timestamps.

Categories and Subject Descriptors: F.3.1 [**Logics and Meanings of Programs**]: Verifying and Reasoning about Programs

General Terms: Security

Additional Key Words and Phrases: formal proofs, security protocols, symbolic constraints, verification

1. INTRODUCTION

Security protocols are small programs that aim at securing communications over a public network, like Internet. Considering the increasing size of networks and their dependence on cryptographic protocols, a high level of assurance is needed in the correctness of such protocols. The design of such protocols is difficult and error-prone; many attacks are dis-

This work has been partially supported by the ACI-SI Satin and the ARA SSIA Formacrypt.

covered even several years after the publication of a protocol. Consequently, there has been a growing interest in applying formal methods for validating cryptographic protocols and many results have been obtained. The main advantage of this approach is its relative simplicity which makes it amenable to automated analysis. For example, the secrecy preservation is co-NP-complete for a bounded number of sessions [Amadio and Lugiez 2000; Rusinowitch and Turuani 2001], and decidable for an unbounded number of sessions under some additional restrictions [Comon-Lundh and Cortier 2003; Durgin et al. 1999; Lowe 1998; Ramanujam and Suresh 2005]. Many tools have also been developed to automatically verify cryptographic protocols, like [Armando et al. 2005; Blanchet 2001; Millen and Shmatikov 2001; Cremers 2008].

Generalizing the constraint system approach. In this paper, we re-investigate and extend the NP-complete decision procedure for a bounded number of sessions [Rusinowitch and Turuani 2001]. In this setting (i.e. finite number of sessions), deducibility constraint systems have become the standard model for verifying security properties, with a special focus on secrecy. Starting with Millen and Shmatikov's paper [Millen and Shmatikov 2001] many results (e.g. [Comon-Lundh and Shmatikov 2003; Baudet 2005; Bursuc et al. 2007]) have been obtained and several tools (e.g. [Corin and Etalle 2002]) have been developed within this framework. Our first contribution is to provide a generic approach derived from [Comon-Lundh and Shmatikov 2003] to decide general security properties. We show that any deducibility constraint system can be transformed in (possibly several) much simpler deducibility constraint systems that are called *solved forms*, preserving *all* solutions of the original system, and not only its satisfiability. In other words, the deducibility constraint system represents in a symbolic way all the possible sequences of messages that are produced, following the protocol rules, whatever are the intruder's actions. This set of symbolic traces is infinite in general. Solved forms are a simple (and finite) representation of such traces and we show that it is suitable for the verification of many security properties. We also consider sorted terms, symmetric and asymmetric encryption, pairing and signatures, but we do not consider algebraic properties like Abelian groups or exclusive or. In addition, we prove termination in *polynomial time* of the (non-deterministic) deducibility constraint simplification. Compared to [Rusinowitch and Turuani 2001], our procedure preserves all solutions. Hence, we can represent for instance, all attacks on the secrecy and not only decide if there exists one. Moreover, presenting the decision procedure using a small set of simplification rules yields more flexibility for further extensions and modifications.

The main originality is that the method is applicable to any security property that can be expressed as a formula on the protocol trace and the agent memories. For example, our decision procedure (published in the LPAR'06 proceedings [Cortier and Zălinescu 2006]) has been used in [Cortier et al. 2006] for proving that a new notion of secrecy in presence of hashes is decidable (and co-NP-complete) for a bounded number of sessions. It has also been used in [Cortier et al. 2007] in the proof of modularity results for security of protocols. To illustrate the large applicability of our decision procedure, we show in this paper how it can be used for proving co-NP-completeness of three kinds of security properties: the existence of key cycles, authentication-like properties, and secrecy of protocols with timestamps.

For authentication properties, we introduce a small logic that allows to specify authentication and some similar security properties. Using our solved forms, we show that any

property that can be expressed within this logic can be decided. The logic is smaller than NPATRL [Syverson and Meadows 1996] or $\mathcal{PS}$-LTL [Corin et al. 2005; Corin 2006], but we believe that decidability holds for a larger logic, closer to the two above ones. However, the goal of this work is not to introduce a new logic, but rather to highlight the proof method. Note also that the absence of key cycles cannot be expressed in any of the three mentioned logics because it is not only a trace property but also a property of the message structure (see below).

For timestamps, we actually retrieve a significant fragment of the decidable class identified by Bozga *et al* [Bozga et al. 2004]. We believe that our result can lead more easily to an implementation, since we only need to adapt the procedure implemented in AVISPA [Armando et al. 2005], while Bozga *et al* have designed a completely new decision procedure, which *de facto* has not been implemented.

Application to key cycles. Our second main contribution is to use this approach to provide an NP-complete decision procedure for detecting the generation of key cycles during the execution of a protocol, in the presence of an intruder, for a bounded number of sessions. To the best of our knowledge, this problem has not been addressed before. The key cycle problem is a problem that arises from the cryptographic community. Indeed, two distinct approaches for the rigorous design and analysis of cryptographic protocols have been pursued in the literature: the so-called Dolev-Yao, symbolic, or formal approach on the one hand and the cryptographic, computational, or concrete approach on the other hand. In the symbolic approach, messages are modeled as formal terms that the adversary can manipulate using a fixed set of operations. In the cryptographic approach, messages are bit strings and the adversary is an arbitrary probabilistic polynomial-time Turing machine. While results in this model yield strong security guarantees, the proofs are often quite involved and only rarely suitable for automation (see, e.g., [Goldwasser and Micali 1984; Bellare and Rogaway 1993]).

Starting with the seminal work of Abadi and Rogaway [Abadi and Rogaway 2002], recent results investigate the possibility of bridging the gap between the two approaches. The goal is to obtain the best of both worlds: simple, automated security proofs that entail strong security guarantees. The approach usually consists in proving that the Dolev-Yao abstraction of cryptographic primitives is correct as soon as strong enough primitives are used in the implementation. For example, in the case of asymmetric encryption, it has been shown [Micciancio and Warinschi 2004b] that the perfect encryption assumption is a sound abstraction for IND-CCA2, which corresponds to a well-established security level. The perfect encryption assumption intuitively states that encryption is a black-box that can be opened only when one has the inverse key. Otherwise, no information can be learned from a cipher-text about the underlying plain-text.

However, it is not always sufficient to find the right cryptographic hypotheses. Formal models may need to be amended in order to be correct abstractions of the cryptographic models. A widely used requirement is to control how keys can encrypt other keys. In a passive setting, soundness results [Abadi and Rogaway 2002; Micciancio and Warinschi 2004a] require that no *key cycles* can be generated during the execution of a protocol. Key cycles are messages like $\mathrm{enc}(k, k)$ or $\mathrm{enc}(k_1, k_2), \mathrm{enc}(k_2, k_1)$ where a key encrypts itself or more generally when the encryption relation between keys contains a cycle. Such key cycles have to be disallowed simply because usual security definitions for encryption schemes do not yield any guarantees otherwise. In the active setting, the typical hypotheses

are even stronger. For instance, in [Backes and Pfitzmann 2004; Janvier et al. 2005] the authors require that a key k never encrypts a key generated before k or, more generally, that it is known in advance which key encrypts which one. More precisely, the encryption relation has to be compatible with the order in which keys are generated, or more generally, it has to be compatible with an a priori given *ordering on keys*.

Related work on key cycles. Some authors circumvent the problem of key cycles by providing new security definitions for encryption, *Key Dependent Messages* security, or KDM in short, that allow key cycles [Adão et al. 2005; Backes et al. 2007]. However, the standard security notions do not imply these new definitions, and ad-hoc encryption schemes have to be constructed. Most of these constructions use the random oracle model, which is provably non implementable. Though there was some recent progress [Hofheinz and Unruh 2008] towards constructing a KDM-secure encryption scheme in the standard model, none of the usual, implemented encryption schemes has been proved to satisfy KDM-security.

In a passive setting, Laud [Laud 2002] proposed a modification of the Dolev-Yao model such that the new model is a sound abstraction even in the presence of key cycles. In his model the intruder's power is strengthened by adding new deduction rules. With the new rules, from a message containing a key cycle, the intruder can infer all keys involved in the cycle as well as the messages encrypted by these keys. Subsequently, Janvier [Janvier 2006] proved that the intruder deduction problem remains polynomial for the modified deduction system. It was also suggested that this approach can be extended to active intruders and incorporated in existing tools, though, to the best of our knowledge, this has not been completed yet. Note that the definition of key cycles used in [Janvier 2006] is more permissive than in [Abadi and Rogaway 2002] (which is unnecessarily restrictive) and it corresponds to the approach of Laud [Laud 2002].

Deciding key cycles. In this paper, we provide an NP-complete decision procedure for detecting the generation of key cycles during the execution of a protocol, in the presence of an active intruder, for a bounded number of sessions. Our procedure works for all the above mentioned definitions of key cycles: strict key cycles (*à la* Abadi, Rogaway), non-strict (*à la* Laud) key cycles, key orderings (*à la* Backes). We therefore provide a necessary component for automated tools used in proving strong, cryptographic security properties, using existing soundness results. Since our approach is an extension of the transformation rules derived from the result of [Rusinowitch and Turuani 2001], we believe that our algorithm can be easily implemented since it can be adapted from the associated procedure, already implemented in AVISPA [Armando et al. 2005] for deciding secrecy and authentication properties.

Outline of the paper. The messages and the intruder capabilities are modeled in Section 2. In Section 3.1, we define deducibility constraint systems and show how they can be used to express protocol executions. In Section 3.2, we define security properties and their satisfaction. In Section 4, we show that the satisfaction of any (in)security property can be non-deterministically, polynomially reduced to the satisfiability of the same problem, this time on simpler constraint systems. The simplification rules derived from [Comon-Lundh and Shmatikov 2003] are provided in Section 4.1. They are actually not sufficient to ensure termination in polynomial time. Thus we introduce in Section 4.6 a refined decision procedure, which is correct, complete, and terminating in polynomial time. We show in

Section 5 how this approach can be used to obtain our main result of NP-completeness for the decision of the key cycles generation. In Section 6, we introduce a small logic to express authentication-like properties and we show how our technique can be used to decide any formula of this logic. In Section 7, we show how it can be used to derive NP-completeness for protocols with timestamps. Some concluding remarks about further work can be found in Section 8.

2. MESSAGES AND INTRUDER CAPABILITIES

2.1 Syntax

Cryptographic primitives are represented by function symbols. More specifically, we consider a *signature* $(\mathcal{S}, \mathcal{F})$ consisting in a set of *sorts* $\mathcal{S} = \{s, s_1 \ldots\}$ and a set of *function symbols* $\mathcal{F} = \{\mathrm{enc}, \mathrm{enca}, \mathrm{sign}, \langle\,\rangle, \mathrm{priv}\}$. Each function symbol is associated with an *arity*: ar is a mapping from $\mathcal{F}$ to $\mathcal{S}^* \times \mathcal{S}$, which we write $\mathrm{ar}(f) = s_1 \times \cdots \times s_n \to s$. The four first function symbols in $\mathcal{F}$ are binary: for each of them there are $s_1, s_2, s \in \mathcal{S}$ such that $\mathrm{ar}(f) = s_1 \times s_2 \to s$. The last symbol is unary: there are $s, s' \in \mathcal{S}$ such that $\mathrm{ar}(f) = s \to s'$.

The symbol $\langle\,\rangle$ represents the pairing function. The terms $\mathrm{enc}(m, k)$ and $\mathrm{enca}(m, k)$ represent respectively the message m encrypted with the symmetric (resp. asymmetric) key k. The term $\mathrm{sign}(m, k)$ represents the message m signed by the key k. The term $\mathrm{priv}(a)$ represents the private key of the agent a. For simplicity, we confuse the agents names with their public key. (Or conversely, we claim that agents identities are defined by their public keys).

$\mathcal{N} = \{a, b \ldots\}$ is a set of *names* and $\mathcal{X} = \{x, y \ldots\}$ is a set of *variables*. Each name and each variable is associated with a sort. We assume that there are infinitely many names and infinitely many variables of each sort.

The set of *terms of sort* s is defined inductively by

t ::=		term of sort s
|	x	variable x of sort s
|	a	name a of sort s
|	$f(t_1, \ldots, t_n)$	application of symbol $f \in \mathcal{F}$ such that $\mathrm{ar}(f) = s_1 \times \cdots \times s_n \to s$ and each t_i is a term of sort s_i.

We assume a special sort Msg that subsumes all the other sorts: any term is of sort Msg.

Sorts are mostly left unspecified in this paper. They can be used in applications to express that certain operators can be applied only to some restricted terms. For example, we use sorts explicitly to express that messages are encrypted by atomic keys (only in Section 5), and to represent timestamps (only in Section 7).

As usual, we write $\mathcal{V}(t)$ for the set of variables occurring in t. For a set T of terms, $\mathcal{V}(T)$ denotes the union of the variables occurring in the terms of T. A term t is *ground* or *closed* if and only if $\mathcal{V}(t) = \emptyset$. A *position* or an *occurrence* in a term t is a sequence of positive integers corresponding to paths starting from the root in the tree-representation of t. For a term t and a position p in this term, $t|_p$ denotes the subterm of t at position p. We write $St(t)$ and $St(T)$ for the set of subterms of a term t, and of a set of terms T, respectively. The *size* of a term t, denoted $|t|$, is defined inductively as usual: $|t| = 1$ if t is a variable or a name and $t = 1 + \sum_{i=1}^{n} |t_i|$ if $t = f(t_1, \ldots, t_n)$ for $f \in \mathcal{F}$. If T is a set of terms then $|T|$ denotes the sum of the sizes of its elements. The cardinality of a set T is denoted

$$\text{Pairing}\quad \frac{S \vdash x \quad S \vdash y}{S \vdash \langle x, y\rangle} \qquad \text{Symmetric encryption}\quad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathrm{enc}(x, y)}$$

$$\text{Asymmetric encryption}\quad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathrm{enca}(x, y)} \qquad \text{Signing}\quad \frac{S \vdash x \quad S \vdash y}{S \vdash \mathrm{sign}(x, y)}$$

$$\text{Symmetric decryption}\quad \frac{S \vdash \mathrm{enc}(x, y) \quad S \vdash y}{S \vdash x} \qquad \text{First Projection}\quad \frac{S \vdash \langle x, y\rangle}{S \vdash x}$$

$$\text{Asymmetric decryption}\quad \frac{S \vdash \mathrm{enca}(x, y) \quad S \vdash \mathrm{priv}(y)}{S \vdash x} \qquad \text{Second Projection}\quad \frac{S \vdash \langle x, y\rangle}{S \vdash y}$$

$$\text{Unsigning}\textit{(optional)}\quad \frac{S \vdash \mathrm{sign}(x, y)}{S \vdash x} \qquad \text{Axiom}\quad \frac{}{S, x \vdash x}$$

Fig. 1. Intruder deduction system.

by $\sharp T$. By abuse of notation, we sometimes denote by T, u the set $T \cup \{u\}$.

Substitutions are written $\sigma = \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$ with $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$. We only consider *well-sorted* substitutions, for which x_i and t_i have the same sort. σ is *closed* if and only if every t_i is closed. The application of a substitution σ to a term t is written $\sigma(t)$ or $t\sigma$. A most general unifier of two terms u and v is denoted by $\mathrm{mgu}(u, v)$.

2.2 Intruder capabilities

The ability of the intruder is modeled by the deduction rules displayed in Figure 1 and corresponds to the usual Dolev-Yao rules.

Pairing, signing, symmetric and asymmetric encryption are the *composition* rules. The other rules are *decomposition rules*. Intuitively, these deduction rules say that an intruder can compose messages by pairing, encrypting, and signing messages provided she has the corresponding keys and conversely, she can decompose messages by projecting or decrypting provided she holds the decryption keys. For signatures, the intruder is also able to *verify* whether a signature $\mathrm{sign}(m, k)$ and a message m match (provided she has the verification key), but this does not give rise to any new message: this capability needs not to be represented in the deduction system. We also consider an optional rule

$$\frac{S \vdash \mathrm{sign}(x, y)}{S \vdash x}$$

that expresses the ability to retrieve the whole message from its signature. This property may or may not hold depending on the signature scheme, and that is why this rule is optional. Note that this rule is necessary for obtaining soundness properties w.r.t. cryptographic digital signatures. Our results will hold in both cases, whether or not this rule is considered in the deduction relation.

A *proof tree* (sometimes simply called a proof) is a tree whose labels are sequents $T \vdash u$ where T is a finite set of terms and u is a term. A proof tree is inductively defined as follows:

—if u is a term and $u \in T$, then $T \vdash u$ is a proof tree whose conclusion is $T \vdash u$, using the axiom;

—if $\pi_1, \ldots, \pi_n$ are proof trees, whose respective conclusions are $T \vdash u_1, \ldots, T \vdash u_n$ respectively and $\dfrac{S \vdash t_1 \quad \cdots \quad S \vdash t_n}{S \vdash t}$ is a rule R of the Figure 1 such that, for some (well-sorted) substitution σ, $t_1\sigma = u_1, \ldots, t_n\sigma = u_n$, then $\dfrac{\pi_1 \quad \cdots \quad \pi_n}{T \vdash t\sigma}$ is a proof tree using R, whose conclusion is $T \vdash t\sigma$.

We will call *subproof* a subtree of a proof tree. An *strict subproof* (resp. *immediate subproof*) of π is a subproof of π distinct from π (resp. a maximal strict subproof of π).

A term u is *deducible* from a set of terms T, which we sometimes write $T \vdash u$ by abuse of notation, if there exists a proof tree whose conclusion is $T \vdash u$.

Example 2.1. The term $\langle k_1, k_2 \rangle$ is deducible from the set $S_1 = \{\mathrm{enc}(k_1, k_2), k_2\}$, as the following proof tree shows:

$$\dfrac{\dfrac{S_1 \vdash \mathrm{enc}(k_1, k_2) \quad S_1 \vdash k_2}{S_1 \vdash k_1} \qquad S_1 \vdash k_2}{S_1 \vdash \langle k_1, k_2 \rangle}$$

3. DEDUCIBILITY CONSTRAINT SYSTEMS AND SECURITY PROPERTIES

Deducibility constraint systems are quite common (see e.g. [Millen and Shmatikov 2001; Comon-Lundh and Shmatikov 2003]) in modeling security protocols. We recall here their definition and show how they can be used to specify general security properties. Then we prove that any deducibility constraint system can be transformed into simpler ones, called *solved*. Such simplified constraints are then used to decide the security properties.

3.1 Deducibility constraint systems

In the usual attacker's model, the intruder controls the network. In particular she can schedule the messages. Once such a scheduling is fixed, she can still replace the messages with fake ones, which are nevertheless accepted by the honest participants. More precisely, some pieces of messages cannot be analyzed by the participants, hence can be replaced by any other piece, provided that the attacker can construct the overall message. This can be used to mount attacks.

In the formal model, pieces that cannot be analyzed are replaced with variables. Any substitution of these variables will be accepted, provided that the attacker can deduce (using the deduction system of Figure 1) the corresponding instance. The main problem then is to decide whether there is such a substitution, yielding a violation of the security property.

Let us give a detailed example recalling how possible execution traces are formalized.

Example 3.1. Consider the famous Needham-Schroeder asymmetric key authentication protocol [Needham and Schroeder 1978] designed for mutual authentication:

$$\begin{array}{ll} A \to B: & \mathrm{enca}(\langle N_A, A \rangle, B) \\ B \to A: & \mathrm{enca}(\langle N_A, N_B \rangle, A) \\ A \to B: & \mathrm{enca}(N_B, B) \end{array}$$

The agent A sends to B his name and a fresh nonce (a randomly generated value) encrypted with the public key of B. The agent B answers by copying A's nonce and adds a fresh

nonce N_B, encrypted by A's public key. The agent A acknowledges by forwarding B's nonce encrypted by B's public key.

Formally, this protocol can be described using two roles A and B. The role A has two parameters: a, b (initiator and responder), and is (informally) specified as follows:

$$\begin{array}{rl} A(a,b) : & \mathsf{generate}(n_a) \\ & A1.\ \mathsf{send}(\mathrm{enca}(\langle n_a, a\rangle, b)) \\ & A2.\ \mathsf{receive}(\mathrm{enca}\,\langle n_a, y\rangle, a) \rightarrow \mathsf{send}(\mathrm{enca}(y, b)) \end{array}$$

where y is a variable: a cannot check that this piece of the message is a nonce generated by b. Hence it can be replaced by any term (or any term of a given sort, depending on what we want to model).

Similarly, the role of B takes the two parameters b, a, and is specified as:

$$\begin{array}{rl} B(b,a) : & \mathsf{generate}(n_b) \\ & B1.\ \mathsf{receive}(\mathrm{enca}(\langle x, a\rangle, b)) \rightarrow \mathsf{send}(\mathrm{enca}(\langle x, n_b\rangle, a)) \\ & B2.\ \mathsf{receive}(\mathrm{enca}(n_b, b)) \end{array}$$

Without loss of generality, we may assume that send actions are performed as soon as the corresponding receive action is completed: this is the best scheduling strategy for the attacker, who will get more information for further computing fake messages. For this reason, we only need to consider the possible scheduling of receive events.

Let a, b be honest participants and i be a corrupted one. Consider one session $A(a, i)$ and one session $B(b, a)$. There are three message deliveries to schedule: $A2, B1, B2$ and $B2$ has to occur after $B1$. Assume the chosen scheduling is $B1, A2, B2$. In this scenario, the possible sequences of message delivery are instances of $\mathrm{enca}(\langle x, a\rangle, b)$, $\mathrm{enca}(\langle n_a, y\rangle, a)$, $\mathrm{enca}(n_b, b)$. The variables x, y can be replaced by any term, provided that the attacker can build the corresponding instances from her knowledge at the appropriate control point.

The initial intruder knowledge can be set to $T_0 = \{a, b, i, \mathrm{priv}(i)\}$, including the private key of the corrupted agent.

For the first message delivery, the attacker has to be able to build the first message instance from this initial knowledge and the message sent at step $A1$:

$$T_1 \stackrel{\mathsf{def}}{=} T_0 \cup \{\mathrm{enca}(\langle n_a, a\rangle, i)\} \Vdash \mathrm{enca}(\langle x, a\rangle, b) \tag{1}$$

This notation will be formally defined later on. Informally, this is a formula, which is satisfied by a substitution σ on x if $\mathrm{enca}(\langle x, a\rangle, b)\sigma$ is deducible from T_1, expressing the ability of the intruder to construct $\mathrm{enca}(\langle x, a\rangle, b)\sigma$.

Then, the agent b replies sending the corresponding instance $\mathrm{enca}(\langle x, n_b\rangle, a)$, which increases the attacker's knowledge, hence enabling its use for building the next message; we get the second deducibility constraint:

$$T_2 \stackrel{\mathsf{def}}{=} T_1 \cup \{\mathrm{enca}(\langle x, n_b\rangle, a)\} \Vdash \mathrm{enca}(\langle n_a, y\rangle, a) \tag{2}$$

Similarly, we construct a third deducibility constraint for the last message delivery:

$$T_3 \stackrel{\mathsf{def}}{=} T_2 \cup \{\mathrm{enca}(y, i)\} \Vdash \mathrm{enca}(n_b, b) \tag{3}$$

Definition 3.2. A *deducibility constraint system* C is a finite set of expressions $T \Vdash u$, called *deducibility constraints*, where T is a non empty set of terms, called the *left-hand side* of the deducibility constraint and u is a term, called the *right-hand side* of the deducibility constraint, such that:

(1) the left-hand sides of all deducibility constraints are totally ordered by inclusion;
(2) if $x \in \mathcal{V}(T)$ for some $(T \Vdash u) \in C$ then

$$T_x \stackrel{\mathsf{def}}{=} \min\{T' \mid (T' \Vdash u') \in C, x \in \mathcal{V}(u')\}$$

exists and $T_x \subsetneq T$.

Informally, the first condition states that the intruder knowledge is always increasing. The second condition expresses that variables abstract pieces of *received* messages: they have to occur first on the right side of a constraint $T \Vdash u$, before occurring in some left side. Note that, due to point (1), T_x exists if and only if the set $\{T' \mid (T' \Vdash u') \in C, x \in \mathcal{V}(u')\}$ is not empty. The linear ordering on left hand sides also implies the uniqueness of the minimum. Hence (2) can be restated equivalently as:

$$(2)\ \ \forall x \in \mathcal{V}(C),\ \exists\, (T \Vdash u) \in C,\ x \in \mathcal{V}(u) \setminus \mathcal{V}(T)$$

In what follows, we may use this formulation instead.

The *left-hand side* of a deducibility constraint system C, denoted by $\mathsf{lhs}(C)$, is the maximal left-hand side of the deducibility constraints of C. The *right-hand side* of a deducibility constraint system C, denoted by $\mathsf{rhs}(C)$, is the set of right-hand sides of its deducibility constraints. $\mathcal{V}(C)$ denotes the set of variables occurring in C. $\bot$ denotes the unsatisfiable system. The *size* of a constraint system is defined as $|C| \stackrel{\mathsf{def}}{=} |\mathsf{lhs}(C) \cup \mathsf{rhs}(C)|$.

A deducibility constraint system C is also written as a conjunction of deducibility constraints

$$C = \bigwedge_{1 \leq i \leq n} (T_i \Vdash u_i)$$

with $T_i \subseteq T_{i+1}$, for all i with $1 \leq i \leq n-1$. The second condition in Definition 3.2 then implies that if $x \in \mathcal{V}(T_i)$ then $\exists j < i$ such that $T_j = T_x$ and $T_j \subsetneq T_i$.

Definition 3.3. A *solution* σ of a deducibility constraint system C is a (well-sorted) ground substitution whose domain is $\mathcal{V}(C)$ and such that, for every $T \Vdash u \in C$, $T\sigma \vdash u\sigma$.

Example 3.4. Coming back to Example 3.1, the substitution $\sigma_1 = \{^{n_a}/_x, ^{n_b}/_y\}$ is a solution of the deducibility constraint system since

$$\begin{array}{rcl} T_0 \cup \{\mathrm{enca}(\langle n_a, a\rangle, i)\} & \vdash & \mathrm{enca}(\langle x, a\rangle, b)\sigma_1 \\ T_1\sigma_1 \cup \{\mathrm{enca}(\langle x, n_b\rangle, a)\sigma_1\} & \vdash & \mathrm{enca}(\langle n_a, y\rangle, a)\sigma_1 \\ T_2\sigma_1 \cup \{\mathrm{enca}(y, i)\sigma_1\} & \vdash & \mathrm{enca}(n_b, b) \end{array}$$

3.2 Security properties

Deducibility constraint systems represent in a symbolic and compact way a possibly infinite set of traces (behaviors), which depend on the attacker's actions. Security properties are formulas, that are interpreted over these traces.

Definition 3.5. Given a set of predicate symbols together with their interpretation over the set of ground terms, a *(in)security property* is a first-order formula ϕ built on these predicate symbols. A *solution* of ϕ is a ground substitution σ of $\mathcal{V}(\phi)$ such that $\phi\sigma$ is true in the given interpretation. (We also write $\sigma \models \phi$).

If C is a deducibility constraint system and ϕ is a (in)security property, possibly sharing free variables with C, a closed substitution σ from $\mathcal{V}(\phi) \cup \mathcal{V}(C)$ is an *attack for ϕ and C*, if is a solution of both C and ϕ.

Example 3.6. If the security property is simply $\mathbf{true}$ (which is always satisfied) and the only sort is Msg then we find the usual deducibility constraint system satisfaction problem, whose satisfiability is known to be NP-complete [Rusinowitch and Turuani 2003].

Example 3.7. Secrecy can be easily expressed by requiring that the secret data is not deducible from the messages sent on the network. We consider again the deducibility constraint system C_1 defined in Example 3.1. The (in)security property then expresses that n_b is deducible: ϕ is the deducibility constraint $T_3 \Vdash n_b$. Note that we may view a constraint (system) as a first order formula.

Then the substitution $\sigma_1 = \{^{n_a}/_x, ^{n_b}/_y\}$ is an attack for ϕ and C_1 and corresponds to the attack found by G. Lowe [Lowe 1996]. Note that such a deduction-based property can be directly included in the constraint system by adding a deducibility constraint $T_3 \Vdash n_b$.

Example 3.8. Let us show here an example of authentication property. Two agents A and B authenticate on some message m if whenever B finishes a session *believing* he has talked to A then A has indeed finished a session with B and they share the same value for m. Note that the agents A and B have in general a different view of the message m, depending e.g. on which nonces they have generated themselves and on which nonces they have received. If m_A denotes the view of m from A and m_B the view of m from B, then the insecurity property states that there is a trace in which these two messages are distinct.

Back to Example 3.1, consider another scenario with two instances of the role A: $A(a, i)$ and $A(a, b)$ and one instance of the role B: $B(b, a)$. The attacker schedules the communications as in Example 3.1: in particular the expected message delivery in $A(a, b)$ is not scheduled (the message is not delivered). Then the deducibility constraint system C_1' is identical to C_1, except that T_0 is replaced with $T_0' = T_0 \cup \{\mathrm{enca}(\langle n_a', a\rangle, b)\}$. The nonce x received by b should correspond to the nonce n_a' sent by a for b; we consider $m_A = n_a'$, $m_B = x$.

The failure of authentication can be stated as the simple formula $x \neq n_a'$. The substitution σ_1 defined in Example 3.7 is then an attack, since b accepts the nonce n_a instead of n_a': $x\sigma_1 \neq n_a'$.

In Sections 5, 6, 7 we provide with other examples corresponding to time constraints, more general authentication-like properties, or to express that no key cycles are allowed.

4. SIMPLIFYING DEDUCIBILITY CONSTRAINT SYSTEMS

Using simplification rules, solving deducibility constraint systems can be reduced to solving simpler constraint systems that we call solved. One nice property of the transformation is that it works for any security property.

Definition 4.1. A deducibility constraint system is *solved* if it is $\bot$ or each of its constraints are of

the form $T \Vdash x$, where x is a variable.

This definition corresponds to the notion of solved form in [Comon-Lundh and Shmatikov 2003]. Note that the empty deducibility constraint system is solved.

Solved deducibility constraint systems with the single sort Msg are particularly simple in the case of the $\mathbf{true}$ predicate since they always have a solution, as noticed in [Millen and Shmatikov 2001]. Indeed, let T_1 be the smallest (w.r.t. inclusion) left hand side of all constraints of a deducibility constraint system. From Definition 3.2, T_1 is non empty and

$$
\begin{array}{lll}
R_1 & C \wedge T \Vdash u \rightsquigarrow C & \text{if } T \cup \{x \mid (T' \Vdash x) \in C, T' \subsetneq T\} \vdash u \\
R_2 & C \wedge T \Vdash u \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathrm{mgu}(t,u),\ t \in St(T),\ t \neq u,\ t,u \text{ not variables} \\
R_3 & C \wedge T \Vdash u \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathrm{mgu}(t_1,t_2),\ t_1,t_2 \in St(T),\ t_1 \neq t_2,\ t_1,t_2 \text{ not variables} \\
R'_3 & C \wedge T \Vdash u \rightsquigarrow_\sigma C\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathrm{mgu}(t_2,t_3),\ \mathrm{enca}(t_1,t_2), \mathrm{priv}(t_3) \in St(T),\ t_2 \neq t_3,\ t_2 \text{ or } t_3 \text{ (or both) is a variable} \\
R_4 & C \wedge T \Vdash u \rightsquigarrow \bot & \text{if } \mathcal{V}(T,u) = \emptyset \text{ and } T \not\vdash u \\
R_f & C \wedge T \Vdash f(u,v) \rightsquigarrow C \wedge T \Vdash u \wedge T \Vdash v & \text{for } f \in \{\langle\rangle, \mathrm{enc}, \mathrm{enca}, \mathrm{sign}\}
\end{array}
$$

Fig. 2. Simplification rules.

has no variables. Let $t \in T_1$. Then the substitution θ defined by $x\theta = t$ for every variable x is a solution since $T \vdash x\theta = t$ for any constraint $T \Vdash x$ in the solved system.

4.1 Simplification rules

The *simplification rules* we consider are defined in Figure 2. For instance, the rule R_1 removes a redundant constraint, i.e., when it is a logical consequence of smaller constraints. The rule R_3 guesses some identity (confusion) between two sent sub-messages.

All the rules are in fact indexed by a substitution: when there is no index then the identity substitution is implicitly assumed. We write $C \rightsquigarrow^n_\sigma C'$ if there are $C_1, \ldots, C_n$ with $n \geq 1$, $C' = C_n$, $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \ldots \rightsquigarrow_{\sigma_n} C_n$, and $\sigma = \sigma_1\sigma_2\ldots\sigma_n$. We write $C \rightsquigarrow^*_\sigma C'$ if $C \rightsquigarrow^n_\sigma C'$ for some $n \geq 1$, or if $C' = C$ and σ is the identity substitution.

Example 4.2. Let us consider the following deducibility constraint system C:

$$\begin{cases} T_1 \Vdash \langle \mathrm{enca}(x,a),\ \mathrm{enca}(y,a) \rangle \\ T_2 \Vdash k_1 \end{cases}$$

where $T_1 = \{a, \langle \mathrm{enca}(k_1,a), \mathrm{enca}(k_2,a)\rangle\}$ and $T_2 = T_1 \cup \{\mathrm{enc}(y,x)\}$. The deducibility constraint system C can be simplified into a solved form using (for example) the following sequence of simplification rules.

$$C \overset{R_{\langle\rangle}}{\rightsquigarrow} \begin{cases} T_1 \Vdash \mathrm{enca}(x,a) \\ T_1 \Vdash \mathrm{enca}(y,a) \\ T_2 \Vdash k_1 \end{cases} \overset{R_{\mathrm{enca}}}{\rightsquigarrow} \begin{cases} T_1 \Vdash x \\ T_1 \Vdash a \\ T_1 \Vdash \mathrm{enca}(y,a) \\ T_2 \Vdash k_1 \end{cases} \overset{R_1}{\rightsquigarrow} \begin{cases} T_1 \Vdash x \\ T_1 \Vdash \mathrm{enca}(y,a) \\ T_2 \Vdash k_1 \end{cases}$$

since $T_1 \vdash a$. Let $\sigma = \mathrm{mgu}\left(\mathrm{enca}(k_1,a),\ \mathrm{enca}(y,a)\right) = \{^{k_1}/_y\}$. We have

$$\begin{cases} T_1 \Vdash x \\ T_1 \Vdash \mathrm{enca}(y,a) \\ T_2 \Vdash k_1 \end{cases} \overset{R_2}{\rightsquigarrow_\sigma} \begin{cases} T_1 \Vdash x \\ T_1 \Vdash \mathrm{enca}(k_1,a) \\ T_2\sigma \Vdash k_1 \end{cases} \overset{R_1}{\rightsquigarrow} \begin{cases} T_1 \Vdash x \overset{R_1}{\rightsquigarrow} T_1 \Vdash x \\ T_2\sigma \Vdash k_1 \end{cases}$$

since $T_1 \vdash \mathrm{enca}(k_1,a)$ and $T_2\sigma \cup \{x\} \vdash k_1$. Intuitively, it means that any substitution of the form $\{^m/_x, {}^{k_1}/_y\}$ such that m is deducible from T_1 is solution of C.

The simplification rules are correct and complete: a deducibility constraint system C has a solution, which is also a solution of a (in)security property ϕ, if and only if there exists a deducibility constraint system C' in solved form such that $C \rightsquigarrow^*_\sigma C'$ and there is a

solution of both C' and $\phi\sigma$. Note that several simplification rules can possibly be applied to a given deducibility constraint system.

THEOREM 4.3. *Let C be a deducibility constraint system, θ a substitution, and ϕ a (in)security property.*

(*1*) *(Correctness) If $C \rightsquigarrow^*_\sigma C'$ for some deducibility constraint system C' and some substitution σ, and if θ is an attack for $\phi\sigma$ and C', then $\sigma\theta$ is an attack for ϕ and C.*

(*2*) *(Completeness) If θ is an attack for C and ϕ, then there exist a deducibility constraint system C' in solved form and substitutions σ, θ' such that $\theta = \sigma\theta'$, $C \rightsquigarrow^*_\sigma C'$, and θ' is an attack for C' and $\phi\sigma$.*

(*3*) *(Termination) There is no infinite derivation sequence $C \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} C_n \cdots$.*

Theorem 4.3 is proved in Sections 4.2, 4.3, and 4.4.

Getting a polynomial bound on the length of simplification sequences requires however an additional memorization technique. This is explained in Section 4.6.

4.2 Correctness

We first give two simple lemmas.

LEMMA 4.4. *If $T \vdash u$ then $\mathcal{V}(u) \subseteq \mathcal{V}(T)$.*

PROOF. The statement follows by induction on the depth of a proof of $T \vdash u$, observing that no deduction rule introduces new variables. Indeed, $\mathcal{V}(t) \subseteq \bigcup_i \mathcal{V}(t_i)$ for deduction rules of the form

$$\frac{S \vdash t_1 \quad \dots \quad S \vdash t_k}{S \vdash t}$$

with $k > 0$, and $\mathcal{V}(t) \subseteq \mathcal{V}(S)$ for the axiom (that is, if $t \in S$). □

The next lemma shows the "cut elimination" property for the deduction system $\vdash$.

LEMMA 4.5. *If $T \vdash u$ and $T, u \vdash v$ then $T \vdash v$.*

PROOF. Consider a proof π of $T \vdash u$ and a proof π' of $T, u \vdash v$. The tree obtained from π' by

—replacing the nodes $T, u \vdash t$ in π' with $T \vdash t$,
—replacing each new leaf $T \vdash u$ (the old $T, u \vdash u$) with the tree π,

is a proof of $T \vdash v$. □

As a consequence, if $T \subseteq T'$, $T' \vdash v$, and $T \vdash u$, for all $u \in T' \setminus T$, then $T \vdash v$.

We show now that the simplification rules preserve deducibility constraint systems.

LEMMA 4.6. *The simplification rules transform a deducibility constraint system into a deducibility constraint system.*

PROOF. Let C be a deducibility constraint system, $C = \bigwedge_i (T_i \Vdash u_i)$ and $C \rightsquigarrow_\sigma C'$. Since $T_i \subseteq T_{i+1}$ implies $T_i\sigma \subseteq T_{i+1}\sigma$, C' satisfies the first point of the definition of deducibility constraint systems.

We show that C' also satisfies the second point of the definition of deducibility constraint systems. Let $(T' \Vdash u') \in C'$ and $x \in \mathcal{V}(T')$. We have to prove that T'_x exists and $T'_x \subsetneq T'$. We distinguish cases, depending on which simplification rule is applied:

—If the rule R_1 is applied, eliminating the constraint $T \Vdash u$. Then $C' = C \setminus \{T \Vdash u\}$. If $T_x \neq T$ then $T'_x = T_x$ (and thus T'_x exists and $T'_x \subsetneq T'$). Suppose that $T_x = T$. Then there is $(T \Vdash u'') \in C$ such that $x \in \mathcal{V}(u'')$. If $u \neq u''$ then again $T'_x = T_x$ (since $(T'_x \Vdash u'') \in C'$). Finally, suppose that $u = u''$. By the minimality of T, it follows that $x \notin \mathcal{V}(T)$ and $x \notin \{y \mid (T'' \Vdash y) \in C, T'' \subsetneq T\}$. Since $x \in \mathcal{V}(u)$, by Lemma 4.4, $T \cup \{y \mid (T'' \Vdash y) \in C, T'' \subsetneq T\} \not\vdash u$, which contradicts the applicability of rule R_1.

—If one of the rules R_2, R_3 or R'_3 is applied, then, for each constraint $(T'' \Vdash u'') \in C'$, there is a constraint $(T \Vdash u) \in C$ such that $T\sigma = T''$ and $u\sigma = u''$. Consider $(T \Vdash u) \in C$ such that $T\sigma = T'$ and $u\sigma = u'$.

If x is not introduced by σ, then $x \in \mathcal{V}(T)$. Then T_x exists and $T_x \subsetneq T$. Thus $T_x\sigma \subseteq T\sigma$. If $T_x\sigma = T\sigma$, then $x \in \mathcal{V}(T_x)$, which contradicts the minimality of T_x. Thus $T_x\sigma \subsetneq T\sigma$. We also have that $\{T''\sigma \mid (T'' \Vdash u'') \in C, x \in \mathcal{V}(u'')\} \subseteq \{T''\sigma \mid (T''\sigma \Vdash u''\sigma) \in C', x \in \mathcal{V}(u''\sigma)\}$, since, for any term u'', if $x \in \mathcal{V}(u'')$, then $x \in \mathcal{V}(u''\sigma)$. It follows that T'_x exists and $T'_x \subseteq T_x\sigma$. Hence $T'_x \subsetneq T'$.

Otherwise, assume that x is introduced by σ: $\exists y \in \mathcal{V}(T)$ such that $x \in \mathcal{V}(y\sigma)$. Then T_y exists and $T_y \subsetneq T$. Let $Y = \{z \in \mathcal{V}(T) \mid x \in \mathcal{V}(z\sigma)\}$ and let $y_0 \in Y$ be such that $T_{y_0} = \min\{T_y \mid y \in Y\}$. For all $y' \in Y$, we have that

$$\begin{aligned} A &\stackrel{\mathsf{def}}{=} \{T''\sigma \mid (T'' \Vdash u'') \in C', x \in \mathcal{V}(u'')\} \\ &= \{T\sigma \mid (T \Vdash u) \in C, x \in \mathcal{V}(u\sigma)\} \\ &\supseteq \{T\sigma \mid (T \Vdash u) \in C, \exists z \in \mathcal{V}(u), x \in \mathcal{V}(z\sigma)\} \\ &\supseteq \{T\sigma \mid (T \Vdash u) \in C, y' \in \mathcal{V}(u), x \in \mathcal{V}(y'\sigma)\} \\ &= \{T\sigma \mid (T \Vdash u) \in C, y' \in \mathcal{V}(u)\} \stackrel{\mathsf{def}}{=} B_{y'}. \end{aligned}$$

Thus $T'_x = \min A \subseteq \min B_{y'} = T_{y'}\sigma$. From $T_{y_0} \subsetneq T$, we obtain that $T_{y_0}\sigma \subseteq T\sigma$. Suppose, by contradiction, that $T_{y_0}\sigma = T\sigma$. Then $x \in \mathcal{V}(T_{y_0}\sigma)$ (since $x \in \mathcal{V}(T\sigma)$). That is, there exists $z \in \mathcal{V}(T_{y_0})$ such that $x \in \mathcal{V}(z\sigma)$. From condition 2 of Definition 3.2 applied to z, it follows that $T_z \subsetneq T_{y_0}$. As z is in Y, this contradicts the choice of y_0. Thus $T'_x \subseteq T_{y_0}\sigma \subsetneq T\sigma = T'$.

—If the rule R_4 is applied then there is nothing to prove.

—If some rule R_f is applied, then the property is preserved, since, if $x \in \mathcal{V}(u'')$ for some term u'' such that $(T'' \Vdash u'') \in C'$, then there is a term v with $x \in \mathcal{V}(v)$ such that $(T'' \Vdash v) \in C$.

□

LEMMA 4.7 CORRECTNESS. *If $C \rightsquigarrow_\sigma C'$, then for every solution τ for C', $\sigma\tau$ is a solution of C.*

PROOF. If C' is obtained by applying R_1, then we have to prove that $T\tau \vdash u\tau$, where $T \Vdash u$ is the eliminated constraint. We know that $T \cup \{x \mid (T' \Vdash x) \in C, T' \subsetneq T\} \vdash u$. It follows that $T\tau \cup \{x\tau \mid (T' \Vdash x) \in C, T' \subsetneq T\} \vdash u\tau$. All constraints $T' \Vdash x$ in C with $T' \subsetneq T$ are also constraints in C'. Thus, for all such constraints, we have that $T'\tau \vdash x\tau$, and hence $T\tau \vdash x\tau$. Then, from Lemma 4.5, we obtain that $T\tau \vdash u\tau$.

If C' is obtained by applying R_2, R_3 or R'_3, then, for every constraint $T \Vdash u$ of C, $(T\sigma)\tau \vdash (u\sigma)\tau$, hence $T(\sigma\tau) \vdash u(\sigma\tau)$.

If C' is obtained by applying some rule R_f, then we obtain that $T\tau \vdash f(u,v)\tau$ from $T\tau \vdash u\tau$ and $T\tau \vdash v\tau$ by applying the corresponding inference rule (e.g. encryption if $f = \mathrm{enc}$).

Finally, C' cannot be obtained by the rule R_4, since it is satisfiable.

It follows that, in all cases, $\sigma\tau$ satisfies C. □

4.3 Completeness

Let $T_1 \subseteq T_2 \subseteq \cdots \subseteq T_n$. We say that a proof π of $T_i \vdash u$ is *left minimal* if, whenever there is a proof of $T_j \vdash u$ for some $j < i$, then, replacing T_i with T_j in all left members of the labels of π, yields a proof of $T_j \vdash u$. In other words, the left-minimal proofs are those that can be performed in a minimal T_j.

We say that a proof is *simple* if all its subproofs are left minimal and there is no repeated label on any branch. Remark that a subproof of a simple proof is simple.

LEMMA 4.8. *If there is a proof of $T_i \vdash u$, then there is a simple proof of it.*

PROOF. We prove the property by induction on the pair (i, m) (considering the lexicographic ordering), where m is the size of a proof of $T_i \vdash u$.

If $i = 1$ then any (subproof of any) proof of $T_1 \vdash u$ is left minimal and there exists a proof without repeated labels on any path.

If $i > 1$ and there is a $j < i$ such that $T_j \vdash u$, then we apply the induction hypothesis to obtain the existence of a simple proof of $T_j \vdash u$. This proof is also a simple proof of $T_i \vdash u$.

If $i > 1$ and there is no $j < i$ such that $T_j \vdash u$, then we apply the induction hypothesis on the immediate subproofs $\pi_1, \ldots, \pi_n$ of the proof π of $T_i \vdash u$. If the label $T_i \vdash u$ appears in one of the resulting proofs π'_i, then replace π with a subproof of π'_i whose conclusion is $T_i \vdash u$. The new proof does not contain any label $T_i \vdash u$. Otherwise, if π is obtained by applying an inference rule R to $\pi_1, \ldots, \pi_n$, then replace π with the proof obtained by applying R to $\pi'_1, \ldots, \pi'_n$. In both cases the resulting proof and all of its subproofs are left minimal by construction, and hence the resulting proof is simple. □

LEMMA 4.9. *Let C be a deducibility constraint system, θ be a solution of C, T_i be a left hand side of C such that, for any $(T \Vdash v) \in C$, if $T \subsetneq T_i$, then v is a variable. Let u be any term. If there is a simple proof of $T_i\theta \vdash u$, whose last inference rule is a decomposition, then there is a non-variable $t \in St(T_i)$ such that $t\theta = u$.*

PROOF. Consider a simple proof π of $T_i\theta \vdash u$. We may assume, without loss of generality, that i is minimal. Otherwise, we simply replace everywhere in the proof T_i with a minimal T_j such that $T_j\theta \vdash u$ is derivable; by left minimality, we get again a proof tree, whose last inference rule is a decomposition. Such a $T_j \subseteq T_i$ also satisfies the hypotheses of the lemma.

We reason by induction on the depth of the proof π. We make a case distinction, depending on the last rule of π:

The last rule is an axiom. Then $u \in T_i\theta$ and there is $t \in T_i$ (thus $t \in St(T_i)$) such that $t\theta = u$. By contradiction, if t was a variable then $T_t \Vdash w$, with $t \in \mathcal{V}(w)$ is a constraint in C such that $T_t \subsetneq T_i$. Moreover, by hypothesis of the lemma, w must be a variable. Hence $w = t$. Then $T_t\theta \vdash u$, which contradicts the minimality of i.

The last rule is a symmetric decryption.

$$\pi = \frac{\overset{\pi_1}{T_i\theta \vdash \mathrm{enc}(u,w)} \quad \overset{\pi_2}{T_i\theta \vdash w}}{T_i\theta \vdash u}$$

By simplicity, the last rule of π_1 cannot be a composition: $T_i\theta \vdash u$ would appear twice on the same path. Then, by induction hypothesis, there is a non variable $t \in St(T_i)$ such that $t\theta = \mathrm{enc}(u,w)$. It follows that $t = \mathrm{enc}(t', t'')$ with $t'\theta = u$. If t' was a variable, then $T_{t'}\theta \vdash t'\theta$ would be derivable. Hence $T_{t'}\theta \vdash u$ would be derivable, which again contradicts the minimality of i. Hence t' is not variable, as required.

The last rule is an asymmetric decryption, (resp. projection, resp. unsigning). The proof is similar to the above one: by simplicity and by induction hypothesis, there is a non-variable $t \in St(T_i)$ such that $t\theta = \mathrm{enca}(u,v)$ (resp. $t\theta = \langle u, v\rangle$, resp. $t\theta = \mathrm{sign}(u, \mathrm{priv}(v))$). Then $t = \mathrm{enca}(t', t'')$ (resp. $t = \langle t', t''\rangle$, resp. $t = \mathrm{sign}(t, t'')$). $t' \in St(T_i)$, $t'\theta = u$ and, by minimality of i, t' is not a variable.

□

LEMMA 4.10. *Let C be a deducibility constraint system and θ be a solution of C. Let T_i be a left hand side of a constraint in C and u be a term, such that:*

(1) *for any $(T \Vdash v) \in C$, if $T \subsetneq T_i$, then v is a variable;*
(2) *T_i does not contain two distinct non-variable subterms t_1, t_2 with $t_1\theta = t_2\theta$;*
(3) *T_i does not contain two terms $\mathrm{enca}(t_1, x)$ and $\mathrm{priv}(t_2)$ where x is a variable distinct from t_2;*
(4) *T_i does not contain two terms $\mathrm{enca}(t_1, t_2)$ and $\mathrm{priv}(x)$ where x is a variable distinct from t_2;*
(5) *u is a non-variable subterm of T_i;*
(6) *$T_i\theta \vdash u\theta$.*

Then $T_i' \vdash u$, where $T_i' = T_i \cup \{x \mid (T \Vdash x) \in C, T \subsetneq T_i\}$.

PROOF. Let j be minimal such that $T_j\theta \vdash u\theta$. Thus $j \leq i$ and $T_j \subseteq T_i$. Consider a simple proof π of $T_j\theta \vdash u\theta$. We reason by induction on the depth of π. We analyze the different cases, depending on the last rule of π:

The last rule is an axiom. Suppose, by contradiction, that $u \notin T_j$. Then there is $t \in T_j$ such that $t\theta = u\theta$ and $t \neq u$. By hypothesis 5, u is not a variable and, by hypothesis 2 of the lemma, t, u cannot be both non-variable subterms of T_i. It follows that t is a variable. Then $T_t\theta \vdash t\theta$, which implies $T_t\theta \vdash u\theta$, contradicting the minimality of j, since $T_t \subsetneq T_j$. Hence $u \in T_j$ and then $T_i' \vdash u$, as required.

The last rule is the symmetric decryption rule. There is w such that $T_j\theta \vdash \mathrm{enc}(u\theta, w)$, $T_j\theta \vdash w$:

$$\frac{T_j\theta \vdash \mathrm{enc}(u\theta, w) \quad T_j\theta \vdash w}{T_j\theta \vdash u\theta}$$

By simplicity, the last rule of the proof of $T_j\theta \vdash \mathrm{enc}(u\theta, w)$ is a decomposition. By Lemma 4.9, there is $t \in St(T_j)$, t not a variable, such that $t\theta = \mathrm{enc}(u\theta, w)$. Let $t = \mathrm{enc}(t_1, t_2)$ and $t_1\theta = u\theta$, $t_2\theta = w$. By induction hypothesis, $T_i' \vdash t$.

If t_1 was a variable, then $T_{t_1} \subsetneq T_j$ and, by hypothesis 1 of the lemma, T_{t_1} must be the left-hand-side of a solved constraint: $(T_{t_1} \Vdash t_1) \in C$ and therefore $T_{t_1}\theta \vdash u\theta$, contradicting the minimality of j.

Now, by hypothesis 5 of the lemma, u is a non-variable subterm of T_i, hence t_1, u are two non variable subterms of T_i such that $t_1\theta = u\theta$. By hypothesis 2 of the lemma, this implies $t_1 = u$.

On the other hand, if t_2 is a variable, $t_2 \in \mathcal{V}(T_i)$ implies $T_{t_2} \subsetneq T_i$ and, since T_i is minimal unsolved, $(T_{t_2} \Vdash t_2) \in C$, which implies $t_2 \in T'_i$. If t_2 is not a variable, then, from $T_j\theta \vdash t_2\theta$ and by induction hypothesis, $T'_i \vdash t_2$. So, in any case, $T'_i \vdash t_2$.

Now, we have both $T'_i \vdash \mathrm{enc}(u, t_2)$ and $T'_i \vdash t_2$, from which we conclude that $T'_i \vdash u$, by symmetric decryption.

The last rule is an asymmetric decryption rule. There is a w such that $T_j\theta \vdash \mathrm{priv}(w)$ and $T_j\theta \vdash \mathrm{enca}(u\theta, w)$. As in the previous case, there is a non-variable $t \in St(T_j)$ such that $t\theta = \mathrm{enca}(u\theta, w)$. By induction hypothesis, $T'_i \vdash t$. Let $t = \mathrm{enca}(t_1, t_2)$.

As in the previous case, t_1 cannot be a variable. Therefore t_1, u are two non-variable subterms of T_i such that $t_1\theta = u\theta$, which implies that $t_1 = u$. (We use here the hypotheses 2 and 5).

On the other hand, the last rule in the proof of $T_j\theta \vdash \mathrm{priv}(w)$ is a decomposition (no composition rule can yield a term headed with priv). Then, by Lemma 4.9 (T_j satisfies the hypotheses of the lemma since $T_j \subseteq T_i$), there is a non-variable subterm $w_1 \in St(T_j)$ such that $w_1\theta = \mathrm{priv}(w)$. Let $w_1 = \mathrm{priv}(w_2)$. By induction hypothesis, $T'_j \vdash \mathrm{priv}(w_2)$.

$$\dfrac{\begin{array}{cc} \mathrm{enca}(t_1,t_2)\theta & \mathrm{priv}(w_2)\theta \\ \| & \| \\ T_j\theta \vdash \mathrm{enca}(u\theta, w) & T_j\theta \vdash \mathrm{priv}(w) \end{array}}{T_j\theta \vdash u\theta}$$

By hypothesis 2 of the lemma, t_2 and w_2 cannot be both non-variable, unless they are identical. Then, by hypotheses 3 and 4 of the lemma, we must have $t_2 = w_2$. Finally, from $T'_i \vdash \mathrm{enca}(u, t_2), T'_i \vdash \mathrm{priv}(t_2)$ we conclude $T'_i \vdash u$.

The last rule is a projection rule.

$$\dfrac{T_j\theta \vdash \langle u\theta, v\rangle}{T_j\theta \vdash u\theta}$$

As before, by simplicity, the last rule of the proof of $T_j\theta \vdash \langle u\theta, v\rangle$ must be a decomposition and, by Lemma 4.9, there is a non variable term $t \in St(T_j)$ such that $t\theta = \langle u\theta, v\rangle$. We let $t = \langle t_1, t_2\rangle$. By induction hypothesis, $T'_i \vdash t$.

Now, as in the previous cases, t_1 cannot be a variable, by minimality of T_j and hypothesis 1 of the lemma. Next, by hypotheses 2 and 5, we must have $t_1 = u$. Finally, from $T'_i \vdash \langle u, t_2\rangle$ we conclude $T'_i \vdash u$ by projection.

The last rule is an unsigning rule.

$$\dfrac{T_j\theta \vdash \mathrm{sign}(u\theta, v)}{T_j\theta \vdash u\theta}$$

This case is identical to the previous one.

The last rule is a composition. Assume for example that it is the symmetric encryption rule.

$$\frac{T_j\theta \vdash v_1 \quad T_j\theta \vdash v_2}{T_j\theta \vdash \mathrm{enc}(v_1, v_2)}$$

with $u\theta = \mathrm{enc}(v_1, v_2)$. Since u is not a variable, $u = \mathrm{enc}(u_1, u_2)$, $u_1\theta = v_1$, and $u_2\theta = v_2$. If u_1 (resp. u_2) is a variable then u_1 (resp. u_2) belongs to $\mathcal{V}(T_i)$ since $u \in St(T_i)$. By point 2 of Definition 3.2 and hypothesis 1 of the lemma, $u_1 \in T'_i$ (resp. $u_2 \in T'_i$).

Otherwise, u_1 and u_2 are non-variables. Then, by induction hypothesis, $T'_i \vdash u_1$ and $T'_i \vdash u_2$. Hence in both cases we have $T'_i \vdash u_1$ and $T'_i \vdash u_2$. Thus $T'_i \vdash u$.

The proof is similar for other composition rules.

□

LEMMA 4.11 COMPLETENESS. *If C is an unsolved deducibility constraint system and θ is a solution of C, then there is a deducibility constraint system C', a substitution σ, and a solution τ of C' such that $C \rightsquigarrow_\sigma C'$ and $\theta = \sigma\tau$.*

PROOF. Consider a constraint $T_i \Vdash u_i$ such that, for any $(T \Vdash v) \in C$ such that $T \subsetneq T_i$, v is a variable and assume u_i is not a variable. If C is unsolved, there is such a constraint in C.

Since θ is a solution, $T_i\theta \vdash u_i\theta$. Consider a simple proof of $T_i\theta \vdash u_i\theta$. We distinguish cases, depending on the last rule applied in this proof:

The last rule is a composition. Since u is not a variable, $u = f(u_1, \ldots, u_n)$ and $T_i\theta \vdash u_j\theta$ for every $j = 1, ..., n$. Then we may apply the transformation rule R_f to C, yielding constraints $T_i \Vdash u_j$ in C' for every j. θ is a solution of the resulting deducibility constraint system C' by hypothesis.

The last rule is an axiom or a decomposition. By Lemma 4.9, there is a non-variable term $t \in St(T_i)$ such that $t\theta = u_i\theta$. We distinguish then again between cases, depending on t, u_i:

Case $t \neq u_i$. Then, since t, u_i are both non-variable terms, we may apply the simplification rule R_2 to C: $C \rightsquigarrow_\sigma C'$ where $C' = C\sigma$ and $\sigma = \mathrm{mgu}(t, u_i)$. Furthermore, $t\theta = u_i\theta$, hence (by definition of a mgu) there is a substitution τ such that $\theta = \sigma\tau$. Finally, θ is a solution of C, hence τ is a solution of C'.

Case $t = u_i$. Then $u_i \in St(T_i)$.

(1) If there are two distinct non-variable terms $t_1, t_2 \in St(T_i)$ such that $t_1\theta = t_2\theta$. Then we apply the simplification rule R_3, yielding a deducibility constraint system $C' = C\sigma$. As in the previous case, there is a substitution τ such that $\theta = \sigma\tau$ and τ is a solution of C'.

(2) If there are $\mathrm{enca}(t_1, t_2), \mathrm{priv}(t_3) \in St(T_i)$ such that either t_2 or t_3 is a variable, $t_2 \neq t_3$ and $t_2\theta = t_3\theta$, then we may apply the rule R'_3 and conclude as in the previous case.

(3) Otherwise, we match all hypotheses of Lemma 4.10 and we conclude that $T'_i \vdash u_i$. Then the rule R_1 can be applied to C, yielding a deducibility constraint system, of which θ is again a solution.

□

4.4 Termination

The simplification rules also terminate, whatever strategy is used for their application:

LEMMA 4.12. *The constraint simplification rules of Figure 2 are (strongly) terminating.*

PROOF. Interpret any deducibility constraint system C as a pair of non-negative integers $I(C) = (n, m)$ where n is the number of variables of the system and m is the number of function symbols occurring in the right hand sides of the system (here, we assume no sharing of subterms). If $C \rightsquigarrow_\sigma C'$, then $I(C) >_{lex} I(C')$ where $\geq_{lex}$ is the lexicographic ordering on pairs of integers. Indeed, the first component strictly decreases by applying R_2, R_3, R'_3, and any other rule strictly decreases the second component, while not increasing the first one. The well foundedness of the lexicographic extension of a well-founded ordering implies the termination of any sequence of rules. □

4.5 Proof of Theorem 4.3

Theorem 4.3 follows from Lemmas 4.7, 4.11, and 4.12, by induction on the derivation length, and since deducibility constraint systems on which no simplification rule can be applied must be solved. Note that the extension of the correctness and completness lemmas to security properties is trivial. Indeed, if ϕ is a (in)security property, then θ is a solution of $\phi\sigma$ if and only if $\sigma\theta$ is a solution of ϕ, for any substitutions θ and σ.

4.6 A decision procedure in NP-time

The termination proof of the last section does not provide with tight complexity bounds. In fact, applying the simplification rules may lead to branches of exponential length (in the size of the constraint system). Indeed when applying a simplification rule to a deducibility constraint, the initial constraint is removed from the constraint system and replaced by new constraint(s). But this deducibility constraint may appear again later on, due to other simplification rules. It is the case for example when considering the following deducibility constraint system.

$$\begin{aligned} T_0 &\stackrel{\mathsf{def}}{=} \{\mathrm{enc}(a, k_0)\} &&\Vdash \mathrm{enc}(x_0, k_0)\\ T_1 &\stackrel{\mathsf{def}}{=} T_0 \cup \{\mathrm{enc}(\langle x_0, \langle x_0, a\rangle\rangle, k_1)\} &&\Vdash \mathrm{enc}(x_1, k_1)\\ &\vdots\\ T_n &\stackrel{\mathsf{def}}{=} T_{n-1} \cup \{\mathrm{enc}(\langle x_{n-1}, \langle x_{n-1}, a\rangle\rangle, k_n)\} &&\Vdash \mathrm{enc}(x_n, k_n)\\ T_{n+1} &\stackrel{\mathsf{def}}{=} T_n \cup \{a\} &&\Vdash x_n \end{aligned}$$

The deducibility constraint system C is clearly satisfiable and its size is linear in n. We have that

$$C \rightsquigarrow_\sigma^{2n} \left\{ \begin{array}{rl} T_0 &\Vdash \mathrm{enc}(x_0, k_0)\\ T_{n+1}\sigma &\Vdash x_n\sigma \end{array} \right.$$

with $\sigma(x_{i+1}) = \langle x_i, \langle x_i, a\rangle\rangle$ for $0 \leq i \leq n-1$. This derivation is obtained by applying rule R_2 and then R_1 for each constraint $T_i \Vdash \mathrm{enc}(x_i, k_i)$ with $1 \leq i \leq n$. The rule R_1 cannot be applied to $T_{n+1}\sigma \Vdash x_n\sigma$ since x_0 and the keys k_i are not present in or derivable from $T_{n+1}\sigma$. Note that $\sigma' = \sigma \cup \{{}^{a}/_{x_0}\}$ is a solution of C and can be easily obtained by rule R_2 on the first constraint and then rule R_1 on both constraints.

However, there is a branch of length $3(2^n-1)$ from $T \Vdash x_n\sigma$ leading to $T \Vdash x_0$ (in solved form), where T denotes $T_{n+1}\sigma$. This is easy to see by induction on n. It is true for $n=0$. Then using only the rules $R_{\langle\rangle}$ and R_1, we have

$$T \Vdash x_n\sigma \overset{R_{\langle\rangle}}{\rightsquigarrow} \left\{\begin{array}{l} T \Vdash x_{n-1}\sigma \\ T \Vdash \langle x_{n-1}\sigma, a\rangle \end{array}\right. \rightsquigarrow^m \left\{\begin{array}{l} T \Vdash x_0 \\ T \Vdash \langle x_{n-1}\sigma, a\rangle \end{array}\right. \overset{R_{\langle\rangle}}{\rightsquigarrow}$$
$$\left\{\begin{array}{l} T \Vdash x_0 \\ T \Vdash x_{n-1}\sigma \\ T \Vdash a \end{array}\right. \overset{R_1}{\rightsquigarrow} \left\{\begin{array}{l} T \Vdash x_0 \\ T \Vdash x_{n-1}\sigma \end{array}\right. \rightsquigarrow^m T \Vdash x_0$$

with $m = 3(2^{n-1}-1)$ by induction hypothesis. The length of the branch is $2 \times 3(2^{n-1}-1)+3 = 3(2^n-1)$. This shows that there exist branches of exponential length in the size of the constraint.

We can prove that it is actually not useful to consider deducibility constraints that have already been seen before (like the constraint $T \Vdash x_{n-1}\sigma$ in our example). Thus we memorize the constraints that have already been visited. The constraint simplification rules, instead of operating on a single deducibility constraint system, rewrite a pair of two constraint systems, the second one representing deducibility constraints that have already been processed at this stage: if $C \rightsquigarrow_\sigma C'$, then

$$C; D \ \rightsquigarrow_\sigma \ C' \setminus D; D \cup (C \setminus C')$$

The constraints ("memorized") in D are those which were already analyzed (i.e. transformed or eliminated). The initial constraint system is $C; \emptyset$.

First, memorization indeed prevents from performing several times the same transformation:

LEMMA 4.13. *If C is a deducibility constraint system and $C; \emptyset \rightsquigarrow^*_\sigma C'; D'$ then $C' \cap D' = \emptyset$.*

PROOF.

$$(C' \setminus D) \cap ((C \setminus C') \cup D) = ((C' \setminus D) \cap D) \cup ((C' \setminus D) \cap (C \setminus C')) = \emptyset$$

□

This kind of memorization is correct and complete in a more general setting. We assume in this section that the reader is familiar with the usual notions of first-order formulas, first-order structures, and models of first-order logic.

A *(general) constraint* is a (first-order) formula, together with an interpretation structure S. A *(general) constraint system* C is a finite set of constraints, whose interpretation is the same as their conjunction. If σ is an assignment of the free variables of C to the domain of S, σ is a *solution* of C if $\sigma, S \models C$. In the context of constraint systems, S is omitted: the satisfaction relation $\models$ refers implicitly to S. It is extended, as usual, to entailment: $C \models C'$ if any solution of C is also a solution of C'. We may consider constraints c as singleton constraint systems, and thus write for example $c \models c'$ instead of $\{c\} \models \{c'\}$.

A *(general) constraint system transformation* is a binary relation $\rightsquigarrow$ on constraints such that, for any sequence (finite or infinite) $C_1 \rightsquigarrow \cdots \rightsquigarrow C_n \rightsquigarrow \cdots$, there is an ordering $\geq$ on individual constraints such that, for every i, for every $c \in C_i \setminus C_{i+1}$, we have

$$\{d \in C_{i+1} \mid d < c\} \models c. \tag{4}$$

This expresses the *correctness* of the transformations: only redundant formulas are removed. The ordering needs not to be well-founded.

Our deducibility constraint systems and deducibility constraint simplification rules satisfy these properties. More precisely, we need to consider the substitutions (partial assignments) as part of the constraint system, in order to fit with the above definition: constraint systems come in two parts: a set of deducibility constraints and a set of solved equations, recording the substitution computed so-far. In other words, a sequence of simplification steps $C_0 \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \dots$ can be written as a general transformation sequence $C_0 \rightsquigarrow (C_1 \wedge \sigma_1) \rightsquigarrow (C_2 \wedge \sigma_1 \wedge \sigma_2) \rightsquigarrow \dots$, where substitutions $\{^{t_1}/_{x_1}, \dots, ^{t_n}/_{x_n}\}$ are seen as conjunctions of solved equations $(x_1 = t_1) \wedge \dots \wedge (x_n = t_n)$.

We show next that for any sequence $C_0 \rightsquigarrow_{\sigma_1} C_1 \rightsquigarrow_{\sigma_2} \dots$ of simplification steps there is an ordering $\geq$ on the corresponding general constraints such that (4) holds.

We start by defining the ordering. First, we order the variables by $x > y$ if, for some i, $y \in \mathcal{V}(x\sigma_1 \dots \sigma_i)$. Intuitively, $x > y$ if x is instantiated before y in the considered derivation. Indeed, let i_x be the minimum among all indexes i such that $x\sigma_i \neq x$ if this minimum exists and ∞ otherwise. Then $x > y$ implies that either $i_x < i_y$, or $i_x = i_y$ and $y \in \mathcal{V}(x\sigma_{i_x})$. (Note that in this last case we cannot have both $y \in \mathcal{V}(x\sigma_{i_x})$ and $x \in \mathcal{V}(y\sigma_{i_x})$, by the definition of a mgu.) This observation proves that the relation $>$ on variables is an ordering. Next, we let $(T \Vdash u) > (T' \Vdash u')$ if

—either the multiset of variables occurring in T is strictly larger than the multiset of variables occurring in T'; such multisets are ordered by the multiset extension of the ordering on variables;

—or else the multisets of variables are identical, and $T' \subsetneq T$;

—or else $T = T'$ and the multiset of variables in u is strictly larger than the multiset of variables in u';

—or else, $T = T'$, the multisets of variable are identical and the size of u is strictly larger than the size of u'.

This is an ordering as a lexicographic composition of orderings. Finally, any solved equation (i.e. substitution) is strictly smaller than any deducibility constraint, and equations are not comparable.

The ordering we have just defined could have been used for the termination proof, as it is a well-founded ordering. It will now be considered as the default ordering on constraints, when a derivation sequence is fixed.

This ordering also satisfies the above required hypotheses for general constraint system transformations, as shown by the proof of the following proposition.

PROPOSITION 4.14. *The simplification rules on deducibility constraint systems form a general constraint system transformation.*

PROOF. Let $C_0 \rightsquigarrow_{\sigma_0} C_1 \rightsquigarrow_{\sigma_1} \dots$ be a simplification sequence. We consider the ordering on deducibility constraints (viewed as general constraints) defined above.

We show next that (4) holds. Note that in (4), c cannot be a solved equation, because at each step solved equations $(x = x\sigma_i)$ may be added but no equation is eliminated. Thus

let $(T \Vdash u) \in C_i \setminus C_{i+1}$, for some $i \geq 0$. We need to show that

$$\bigwedge_{\substack{(T' \Vdash u') \in C_{i+1} \\ (T' \Vdash u') < (T \Vdash u)}} T' \Vdash u' \wedge \bigwedge_{1 \leq j \leq i} \sigma_j \quad \models \quad T \Vdash u \tag{5}$$

We investigate the possible transformation rules.

For the rules R_2, R_3, R_3', $C_{i+1} = C_i\sigma_i$. We have $(T \Vdash u) \geq (T\sigma_i \Vdash u\sigma_i)$ since either the multiset of variables of $T\sigma_i$ is strictly smaller than the multiset of variables of T, or else $T = T\sigma_i$ and, in the latter case, either the multiset of variables of $u\sigma_i$ is strictly smaller than the multiset of variables of u or else $u\sigma_i = u$. Moreover, $c\sigma \wedge \sigma \models c$ for all constraints c and substitutions σ. Indeed, if θ is a solution of $c\sigma \wedge \sigma$ then $x\theta = x\sigma\theta$ for any $x \in \mathrm{dom}(\sigma)$. It follows that $c\theta = c\sigma\theta$, and thus θ is a solution of c.

Hence, we have in particular that $(T\sigma_i \Vdash u\sigma_i) \wedge \sigma_i \models T \vdash u$, which shows that (5) holds for this case.

For the rule R_f, it suffices to notice that $\{T \Vdash u_1, \ldots, T \Vdash u_n\} \models (T \Vdash f(u_1, \ldots, u_n))$ and $(T \Vdash u_i) < (T \Vdash f(u_1, \ldots, u_n))$ for every i.

For the rule R_1, the constraint $T \Vdash u$ is a consequence of the (strictly smaller) constraints $T' \Vdash x$ for $T' \subsetneq T$.

Finally, the rule R_4 only applies to unsatisfiable deducibility constraints. □

The memorization strategy can be defined, as above, for any general constraint system transformation. The correctness of the memorization strategy relies on the following invariant:

LEMMA 4.15. *For any constraint system transformation $\rightsquigarrow$, if $C; \emptyset \rightsquigarrow^* C'; D'$, then $C' \models D'$.*

PROOF. We prove, by induction on the length of the derivation sequence the following stronger result: $\forall d \in D', \{c \in C' \mid c < d\} \models d$.

The base case is straightforward as D' is empty. Next, assume that $C; D \rightsquigarrow C'; D'$. By definition, $D' = D \cup (C \setminus C')$. If $d \in C \setminus C'$, by definition of a constraint transformation rule, $\{c \in C' \mid c < d\} \models d$. If $d \in D$, by induction hypothesis, $\{c \in C \mid c < d\} \models d$. Hence $\{c \in C' \,|c < d\} \cup \{c \in C \setminus C' \mid c < d\} \models d$. But, again by definition of constraint transformations, any constraint in the second set is a consequence of the first set: we get $\{c \in C' \,|c < d\} \models d$. □

It follows that the memorization strategy is always correct when the original constraint transformation is correct.

Now, the memorization strategy preserves the properties of our deducibility constraint systems:

LEMMA 4.16. *If C is a deducibility constraint system and $C; \emptyset \rightsquigarrow_\sigma^* C'; D'$ then C' is a deducibility constraint system.*

PROOF. Let $(C_i; D_i) \rightsquigarrow_{\sigma_{i+1}} (C_{i+1}; D_{i+1})$, with $0 \leq i < n$ be the sequence of deducibility constraint systems obtained by applying successively the simplification rules, where $C_0 = C$, $D_0 = \emptyset$, $C_n = C'$, and $C_i \rightsquigarrow_{\sigma_{i+1}} C'_{i+1}$ (and thus $C_{i+1} = C'_{i+1} \setminus D_i$, and $D_{i+1} = D_i \cup (C_i \setminus C'_{i+1})$). We know that C'_i is a deducibility constraint system, by Lemma 4.6.

First, the left members of C_i are linearly ordered by inclusion, as they are a subset of the left members of C'_i.

We consider now the other property of deducibility constraint systems. We let $\geq$ be the ordering on constraints defined before. We show below, by induction on i that, for every $x \in \mathcal{V}(C_i)$, for every $(T \Vdash u) \in D_i$ such that $x \in \mathcal{V}(u) \setminus \mathcal{V}(T)$, there is a $(T' \Vdash u') \in C_i$ such that $x \in \mathcal{V}(u') \setminus \mathcal{V}(T')$ and $(T' \Vdash u') < (T \Vdash u)$.

Note that this property implies that C_i is a deducibility constraint system: For every variable $x \in \mathcal{V}(C_i)$, there is $(T_x \Vdash u) \in C'_i$ such that $x \in \mathcal{V}(u) \setminus \mathcal{V}(T_x)$, as C'_i is a deducibility constraint system. If $(T_x \Vdash u) \in C_i$ then we're done, otherwise $(T_x \Vdash u) \in D_i$, and hence, by the stated property, there is $(T'_x \Vdash u') \in C_i$ such that $x \in \mathcal{V}(u') \setminus \mathcal{V}(T'_x)$. This shows that C_i is a deducibility constraint system.

The property holds trivially for $i = 0$. For the induction step, let $x \in \mathcal{V}(C_{i+1})$ and $(T \Vdash u) \in C'_{i+1}$ be such that $x \in \mathcal{V}(u) \setminus \mathcal{V}(T)$. We investigate three cases:

—if C_{i+1} is obtained by one of the rules R_2, R_3, R'_3, then $C_{i+1} = C_i\sigma_{i+1} \setminus D_i$, and $x \notin \mathrm{dom}(\sigma_{i+1})$. We assume w.l.o.g. that $T \Vdash u$ is a minimal constraint in D_{i+1} such that $x \in \mathcal{V}(u) \setminus \mathcal{V}(T)$.
There is $(T' \Vdash u') \in C_i$ such that $x \in \mathcal{V}(u') \setminus \mathcal{V}(T')$ and $(T' \Vdash u') \leq (T \Vdash u)$: if $(T \Vdash u) \notin C_i$, then $(T \Vdash u) \in D_i$ and by induction hypothesis, there is a $(T' \Vdash u') \in C_i$ such that $x \in \mathcal{V}(u') \setminus \mathcal{V}(T')$ and $(T' \Vdash u') < (T \Vdash u)$.
Let $S = \{y \in \mathcal{V}(T') \mid x \in \mathcal{V}(y\sigma_{i+1})\}$. By induction hypothesis C_i is a constraint system, and hence, for every $y \in S$, there is a (minimal) constraint $T_y \Vdash u_y \in C_i$ such that $y \in \mathcal{V}(u_y) \setminus \mathcal{V}(T_y)$. Since $y \in \mathcal{V}(T')$, $T_y \subsetneq T'$. Let $T_1 \Vdash u_1$ be a minimal element in $\{T_y \Vdash u_y \mid y \in S\} \cup \{T' \Vdash u'\}$. Suppose that $x \in \mathcal{V}(T_1\sigma_{i+1})$. Since $x \notin \mathcal{V}(T')$ and $T_y \subsetneq T'$, it follows that $x \notin \mathcal{V}(T_y)$, and hence there is $z \in \mathcal{V}(T_1)$ such that $x \in \mathcal{V}(z\sigma_{i+1})$. It follows that $z \in S$ and $T_z \subsetneq T_1$, which contradicts the minimality of $T_1 \Vdash u_1$. Hence $x \in \mathcal{V}(u_1\sigma_{i+1}) \setminus \mathcal{V}(T_1\sigma_{i+1})$. Also $(T_1\sigma_{i+1} \Vdash u_1\sigma_{i+1}) \leq (T_1 \Vdash u_1) \leq (T' \Vdash u') \leq (T \Vdash u)$. Furthermore, at least one of the inequalities is strict: if $(T \Vdash u) \in D_i$ the last inequality is strict, otherwise $(T \Vdash u) \in (C_i \setminus C'_{i+1}) = (C_i \setminus C_i\sigma)$ hence $(T\sigma_{i+1} \Vdash u\sigma_{i+1}) < (T \Vdash u)$. It follows that $(T_1\sigma_{i+1} \Vdash u_1\sigma_{i+1}) \in C_{i+1}$ by minimality of $T \Vdash u$.

—if C_{i+1} is obtained by an R_f rule. We may assume w.l.o.g. that $T \Vdash u$ is a minimal constraint in D_{i+1} such that $x \in \mathcal{V}(u) \setminus \mathcal{V}(T)$.
Either $(T \Vdash u) \in D_i$, in which case, by induction hypothesis, there is $(T' \Vdash u') \in C_i$ such that $x \in \mathcal{V}(u') \setminus \mathcal{V}(T')$ and $(T' \Vdash u') < (T \Vdash u)$. If $(T' \Vdash u') \in C_{i+1}$, there is nothing to prove. Otherwise, $u' = f(u_1, \ldots, u_n)$ and, for every j, $(T' \Vdash u_j) \in C_{i+1} \cup D_i$. Moreover, there is an index j such that $x \in \mathcal{V}(u_j) \setminus \mathcal{V}(T')$ and, by minimality of $T \Vdash u$, $(T' \Vdash u_j) \in C_{i+1}$, hence completing this case.
Or else $(T \Vdash u) \in C_i \setminus C'_{i+1}$, in which case $u = f(u_1, \ldots, u_n)$ and $(T \Vdash u_j) \in C_{i+1} \cup D_i$. As above, we conclude that for some j, $x \in \mathcal{V}(u_j) \setminus \mathcal{V}(T)$, $(T \Vdash u_j) \in C_{i+1}$ and $(T \Vdash u_j) < (T \Vdash u)$.

—if C_{i+1} is obtained by the rule R_1, removing a constraint $T_1 \Vdash u_1$, then $D_{i+1} = D_i \cup \{T_1 \Vdash u_1\}$ and, by Lemma 4.6 for any variable $y \in \mathcal{V}(u_1) \setminus \mathcal{V}(T_1)$ there is a strictly smaller constraint $(T_2 \Vdash u_2) \in C_i$ such that $y \in \mathcal{V}(u_2) \setminus \mathcal{V}(T_2)$. Then we simply apply the induction hypothesis.

□

THEOREM 4.17. *Let C be a deducibility constraint system, θ a substitution and ϕ a security property.*

(1) *(Correctness) If $C;\emptyset \rightsquigarrow^*_\sigma C';D'$ for some deducibility constraint system C' and some substitution σ, if θ is an attack for C' and $\phi\sigma$, then $\sigma\theta$ is an attack for C and ϕ.*

(2) *(Completeness) If θ is an attack for C and ϕ, then there exist a deducibility constraint system C' in solved form, a set of deducibility constraints D' and substitutions σ,θ' such that $\theta=\sigma\theta'$, $C;\emptyset \rightsquigarrow^*_\sigma C';D'$, and θ' is an attack for C' and $\phi\sigma$.*

(3) *(Termination) If $C;\emptyset \rightsquigarrow^n_\sigma C';D'$ for some deducibility constraint system C' and some substitution σ, then n is polynomially bounded in the size of C.*

PROOF. For correctness, we rely on Lemmas 4.7, and 4.15: by Lemma 4.15, any solution θ of C' is also a solution $C'\cup D'\sigma$ and, by Lemma 4.7 (and induction), $\sigma\theta$ is a solution of C.

For completeness, from Lemma 4.11, we know that if C_i is an unsolved deducibility constraint system and θ is an attack for C_i and ϕ, then there is a deducibility constraint system C'_{i+1}, a substitution σ_i, and an attack τ_i for C'_{i+1} and $\phi\sigma_i$ such that $C_i \rightsquigarrow_{\sigma_i} C'_{i+1}$ and $\theta=\sigma_i\tau_i$. Then τ_i is an attack also for $C'_{i+1}\setminus D_i$ and $\phi\sigma$, for any set of constraints D_i. By Lemma 4.16, we know that when D_i represents already visited constraints, then $C'_{i+1}\setminus D_i$ is a deducibility constraint system. We can thus conclude by induction on the derivation length n, taking $C_0=C$, $D_0=\emptyset$, $C_{i+1}=C'_{i+1}\setminus D_i$ for all i, and $C_n=C'$.

Concerning termination, we assume a DAG representation of the terms and constraints, in such a way that the size of the constraint is proportional to the number of the distinct subterms occurring in it. Next, observe that $\sharp St(t\sigma) \leq \sharp(St(t)\cup\bigcup_{x\in\mathrm{dom}(\theta)} St(x\theta))$. Hence, when unifying two subterms of t, with $\mathrm{mgu}\ \theta$, $\sharp St(t\theta)\leq\sharp St(t)$ since, for every variable $x\in\mathrm{dom}(\theta)$, $x\theta$ is a subterm of t. It follows that, for any constraint system $C';D'$ such that $C;\emptyset\rightsquigarrow^*_\sigma C';D'$, $\sharp St(C')\leq\sharp St(C)$.

Next, observe that the number of distinct left hand sides of the constraints $\sharp\mathsf{lhs}(C')$ is never increasing: $\sharp\mathsf{lhs}(C')\leq\sharp\mathsf{lhs}(C)$. Furthermore, as long as we only apply the rules R_1,R_f, starting from C'', the left hand sides of the deducibility constraint systems are fixed: there are at most $\sharp\mathsf{lhs}(C'')$ of them. Now, since, thanks to memorization, we cannot get twice the same constraint, the number of consecutive R_1,R_f steps is bounded by

$$\sharp\mathsf{lhs}(C'')\times\sharp St(\mathsf{rhs}(C''))\leq\sharp\mathsf{lhs}(C)\times\sharp St(C)$$

It follows that the length of a derivation sequence is bounded by $\sharp\mathcal{V}(C)\times\sharp\mathsf{lhs}(C)\times\sharp St(C)$ (for R_1,R_f steps) plus $\sharp\mathcal{V}(C)$ (for R_2,R_3,R'_3 steps) plus 1 (for a possible R_4 step). □

Theorem 4.17 extends the result of [Rusinowitch and Turuani 2001] to sorted messages and general security properties. Handling arbitrary security properties is possible as soon as we do not forget any solution of the deducibility constraint systems (as we do). If we only preserve the existence of a solution of the constraint (as in [Rusinowitch and Turuani 2001]), it might be the case that the solution of C that we kept is not a solution of the property ϕ, while there are solutions of both ϕ and C, that were lost in the satisfiability decision of C. In addition, compared to [Rusinowitch and Turuani 2001], presenting the decision procedure using a small set of simplification rules makes it more easily amendable to further extensions and modifications. For example, Theorem 4.17 has been used

in [Cortier et al. 2006] for proving that a new notion of secrecy in presence of hashes is decidable (and co-NP-complete) for a bounded number of sessions.

Note that termination in polynomial time also requires the use of a DAG (Directed Acyclic Graph) representation for terms.

The following corollary is easily obtained from the previous theorem by observing that we can guess the simplification rules which lead to a solved form.

COROLLARY 4.18. *Any property ϕ that can be decided in polynomial time on solved deducibility constraint systems can be decided in non-deterministic polynomial time on arbitrary deducibility constraint systems.*

4.7 An alternative approach to polynomial-time termination

Inspecting the completeness proof, there is still some room for choosing a strategy, while keeping completeness (correctness is independent of the order of the rules application). To obtain even more flexibility, we slightly relax the condition on the application of the rule R_2 on a constraint $T \Vdash u$: we require unifying a subterm $t \in St(T)$ and a subterm $t' \in St(u)$ (instead of unifying t with u) where, as before, $t \neq t'$, t, t' non-variables. Remark that this change preserves the completeness of the procedure.

Let us group the rules R_2, R_3, R'_3 and call them *substitution rules* S. We write $S(u, v)$ if the substitution is obtained by unifying u and v. There are some basic observations:

(1) If $C \rightsquigarrow^{R_f} C' \rightsquigarrow^{S}_{\sigma} C'\sigma$, then $C \rightsquigarrow^{S}_{\sigma} C\sigma \rightsquigarrow^{R_f} C'\sigma$. Hence we may always move forward the substitution rules.

(2) If $C_1 \rightsquigarrow^{R_f} C'_1$ and $C_2 \rightsquigarrow^{R_f} C'_2$, then $C_1 \wedge C_2 \rightsquigarrow^{R_f} C'_1 \wedge C_2 \rightsquigarrow^{R_f} C'_1 \wedge C'_2$ and $C_1 \wedge C_2 \rightsquigarrow^{R_f} C_1 \wedge C'_2 \rightsquigarrow^{R_f} C'_1 \wedge C'_2$, hence any two consecutive applications of R_f on different constraints can be performed in any order.

(3) The rules R_1, R_4 can be applied at any time when they are enabled; we may apply them eagerly or postpone them until no other rule can be applied.

(4) If $C \rightsquigarrow^{S(u_1,v_1)}_{\sigma_1} C\sigma_1 \rightsquigarrow^{S(u_2\sigma_1,v_2\sigma_1)}_{\sigma_2} C\sigma_1\sigma_2$, then, for some θ_1, θ_2,

$$C \rightsquigarrow^{S(u_2,v_2)}_{\theta_1} C\theta_1 \rightsquigarrow^{S(u_1\theta_1,v_1\theta_1)}_{\theta_2} C\sigma_1\sigma_2$$

Hence any two consecutive substitution rules can be performed in any order.

(5) If $C \rightsquigarrow^{S}_{\sigma} C\sigma \rightsquigarrow^{R_f} C'\sigma$, and $S \neq R_2$, then $C \rightsquigarrow^{R_f} C' \rightsquigarrow^{S}_{\sigma} C'\sigma$.

This provides with several complete strategies. For instance the following strategy is complete:

—apply eagerly R_4 and postpone R_1 as much as possible

—apply the substitution rules eagerly (as soon as they are enabled). This implies that all substitution rules are applied at once, since the rules R_1, R_4, R_f cannot enable a substitution.

—when R_4 and substitutions rules are not enabled, apply R_f to the constraint, whose right hand side is maximal (in size).

Such a strategy will also yield polynomial length derivations, since we cannot get twice the same constraint: in any derivation sequence $C_0 \rightsquigarrow_{\sigma_1} \cdots \rightsquigarrow_{\sigma_n} C_n$, if $(T \Vdash u) \in C_i \setminus C_{i+1}$ (we say then that $T \Vdash u$ has been eliminated at this step), then, for any $j > i$, $(T \Vdash u) \notin C_j$. Indeed, for the substitution rules, $T \Vdash u$ is eliminated only when $x \in \mathcal{V}(T \Vdash u)$ and

$x \in \mathrm{dom}(\sigma_{i+1})$, in which case for any $j > i$, $x \notin \mathcal{V}(C_j)$. And, if $T \Vdash u$ is eliminated by an R_f rule, then $|u| = \max_{t \in \mathsf{rhs}(C_i)} |t|$. If, for some $j > i$, the constraint $T \Vdash u$ was in C_{j+1} and not in C_j, then we would have $\max_{t \in \mathsf{rhs}(C_j)} |t| > |u|$. Thus the maximum of the sizes of the right hand sides terms would have increased, which is not possible according to our strategy.

Then the complexity analysis of the proof of Theorem 4.17 can be applied here.

The above observations can also be used to bound the non-determinism (which is useful in practice): for instance from (1) and (4), we see that substitution rules can be applied "don't care": if we use a substitution rule, we do not need to consider other alternatives. More precisely, if $S(t, u)$ is a substitution rule that is applicable to C, let $\Phi(C)$ be the set of substitution rules $S(t', u')$, which are applicable to C and such that there is no θ other than the identity such that $\mathrm{mgu}(t, u)\theta = \mathrm{mgu}(t', u')$. Then

$$\theta \models C \implies \bigvee_{S(t',u') \in \Phi(C)} \exists \theta'.\ \theta = \mathrm{mgu}(t', u')\theta'$$

Similarly, from (5), a right-hand side member that is not unifiable with a non-variable subterm of the corresponding left hand side, can be "don't care" decomposed:

$$\theta \models C \wedge (T \Vdash f(u_1, \ldots, u_n)) \implies \theta \models C \wedge (T \Vdash u_1) \wedge \ldots \wedge (T \Vdash u_n)$$

if $f(u_1, \ldots, u_n)$ is not unifiable with any non-variable subterm of T.

5. DECIDABILITY OF ENCRYPTION CYCLES

Using the general approach presented in the previous section, verifying particular properties like the existence of key cycles or the conformation to an *a priori* given ordering relation on keys can be reduced to deciding these properties on solved deducibility constraint systems. We deduce a new decidability result, useful in models designed for proving cryptographic properties.

To show that formal models (like the one presented in this article) are sound with respect to cryptographic ones, the authors usually assume that no key cycle can be produced during the execution of a protocol or, even stronger, assume that the "encrypts" relation on keys follows an *a priori* given ordering.

For simplicity, and since there are very few papers constraining the key relations in an asymmetric setting, in this section we restrict our attention to key cycles and key orders on symmetric keys. Moreover, we consider atomic keys for symmetric encryption since there exists no general definition (with a cryptographic interpretation) of key cycles in the case of arbitrary composed keys and soundness results are usually obtained for atomic keys.

More precisely, we assume a sort $\mathsf{Key} \subset \mathsf{Msg}$ and we assume that the sort of enc is $\mathsf{Msg} \times \mathsf{Key} \to \mathsf{Msg}$. All the other symbols are of sort $\mathsf{Msg} \times \cdots \times \mathsf{Msg} \to \mathsf{Msg}$. Hence only names and variables can be of sort Key. In this section we call *key* a variable or a name of sort Key. Finally, for any list of terms L, L_s is the set of terms that are members of the list.

In this section, we consider (in)security properties of the form $P(L)$ where P is a predicate symbol and L is a list of terms. Informally, σ will be a solution of $P(L)$ if $L_s\sigma$ contains a key cycle. The precise interpretation of P depends on the notion of key-cycle: this is what we investigate first in the following section.

5.1 Key cycles

Many definitions of key cycles are available in the literature. They are stated in terms of an "encryption" relation between keys or occurrences of keys. An early definition proposed by Abadi and Rogaway [Abadi and Rogaway 2002], identifies a key cycle with a cycle in the encryption relation, with no conditions on the occurrences of the keys. However, the definition induced by Laud's approach [Laud 2002] corresponds to searching for such cycles only in the "visible" parts of a message. For example the message $\mathrm{enc}(\mathrm{enc}(k,k),k')$ contains a key cycle using the former definition but does not when using the latter one and assuming that k' is secret. It is generally admitted that the Abadi-Rogaway definition is unnecessarily restrictive and hence we will say that the corresponding key cycles are *strict*. However, for completeness reasons, we treat both cases.

There can still be other variants of the definition, depending on whether the relation "k encrypts k'" is restricted or not to keys k' that occur in plain-text. For example, $\mathrm{enc}(\mathrm{enc}(a,k),k)$ may or may not contain a key cycle. As above, even if occurrences of keys used for encrypting (as k in $\mathrm{enc}(m,k)$) need not be considered as encrypted keys, and hence can safely be ignored when defining key cycles, we consider both cases. Note that the initial Abadi-Rogaway setting considers that $\mathrm{enc}(\mathrm{enc}(a,k),k)$ has a key cycle.

We write $s <_{st} t$ if and only if s is a subterm of t. $\sqsubseteq$ is the least reflexive and transitive relation satisfying: $s_1 \sqsubseteq (s_1,s_2)$, $s_2 \sqsubseteq (s_1,s_2)$, and, if $s \sqsubseteq t$, then $s \sqsubseteq \mathrm{enc}(t,t')$. Intuitively, $s \sqsubseteq t$ if s is a subterm of t that either occurs (at least once) in clear (i.e. not encrypted) or occurs (at least once) in a plain-text position. A position p is a *plain-text position* in a term u if there exists an occurrence q of an encryption in u such that $q \cdot 1 \leq p$.

Definition 5.1. Let ρ_1 be a relation chosen in $\{<_{st}, \sqsubseteq\}$. Let S be a set of terms and k, k' be two keys. We say that k *encrypts* k' *in* S (denoted $k\, \rho_e^S\, k'$) if there exist $m \in S$ and a term m' such that

$$k'\, \rho_1\, m' \text{ and } \mathrm{enc}(m',k) \sqsubseteq m.$$

For simplicity, we may write ρ_e instead of ρ_e^S, if S is clear from the context. Also, if m is a message we denote by ρ_e^m the relation $\rho_e^{\{m\}}$.

Let S be a set of terms. We define $\mathsf{hidden}(S) \stackrel{\mathsf{def}}{=} \{k \in St(S) \mid k \text{ of sort } \mathsf{Key}, S \not\vdash k\}$.

Definition 5.2 *(Strict key cycle)*. Let K be a set of keys. We say that a set of terms S contains a *strict key cycle* on K if there is a cycle in the restriction of the relation ρ_e^S on K. Otherwise we say that S is *strictly acyclic* on K.

We define the predicate P_{skc} as follows: $L \in P_{skc}$ if and only if the set $\{m \mid L_s \vdash m\}$ contains a strict key cycle on $\mathsf{hidden}(L_s)$.

We give now the definition induced by Laud's approach [Laud 2002]. He has showed in a passive setting that if a protocol is secure when the intruder's power is given by a modified Dolev-Yao deduction system $\vdash_\emptyset$, then the protocol is secure in the computational model, without requiring a "no key cycle" condition. Rephrasing Laud's result in terms of the standard deduction system $\vdash$ gives rise to the definition of key cycles below, as it has been proved in [Janvier 2006].

To state the following definition we need a more precise notion than the encrypts relation. We say that an occurrence q of a key k *is protected* by a key k' in a term m if $m|_{q'} = \mathrm{enc}(m',k')$ for some term m' and some position q', and the occurrence of k at q in m is a plain-text occurrence of k in m', that is $q' \cdot 1 \leq q$. We extend this definition in

the intuitive way to sets of terms. This can be done for example by indexing the terms in the set and adding this index as a prefix to the position in the term to obtain the position in the set.

Definition 5.3 *(Key cycle [Janvier 2006])*. Let K be a set of keys. We say that a set of terms S is *acyclic* on K if there exists a strict partial ordering $\prec$ on K such that for all $k \in K$, for all occurrences q of k in plain-text position in S, there is $k' \in K$ such that $k' \prec k$ and q is protected by k' in S. Otherwise we say that S contains a *key cycle* on K.

We define the predicate P_{kc} as follows: for any list of terms L, $L \in P_{kc}$ if and only if the set $\{m \mid L_s \vdash m\}$ contains a key cycle on $\mathsf{hidden}(L_s)$.

We say that a term m contains a (strict) key cycle if the set $\{m\}$ contains one.

Example 5.4. The messages $m = \mathrm{enc}(\mathrm{enc}(k,k),k')$ and $m' = \langle \mathrm{enc}(k_1,k_2), \mathrm{enc}(\mathrm{enc}(k_2,k_3),k_1)\rangle$ are acyclic, while the message $m'' = \langle\langle \mathrm{enc}(k_1,k_2), \mathrm{enc}(\mathrm{enc}(k_2,k_1),k_3)\rangle, k_3\rangle$ has a key cycle. The orderings $k' \prec k$ and $k_3 \prec k_2 \prec k_1$ prove it for m and m' while for m'' such an ordering cannot be found since k_3 is deducible. However, all three messages have strict key cycles.

5.2 Key orderings

In order to establish soundness of formal models in a symmetric encryption setting, the requirements on the encrypts relation can be even stronger, in particular in the case of an active intruder. In [Backes and Pfitzmann 2004] and [Janvier et al. 2005] the authors require that a key never encrypts a younger key. More precisely, the encrypts relation has to be compatible with the ordering in which the keys are generated. Hence we also want to check whether there exist executions of the protocol for which the encrypts relation is incompatible with an *a priori* given order on keys.

Definition 5.5 *(Key ordering)*. Let $\prec$ be a strict partial ordering on a set of keys K. We say that a set of terms S is *compatible* with $\prec$ on K if

$$k \,\rho_e^S\, k' \;\Rightarrow\; k' \not\preceq k, \text{ for all } k,k' \in K.$$

Given a strict partial ordering $\prec$ on a set of keys, we define the predicate $P_\prec$ as follows: $P_\prec$ holds on a list of terms L if and only if the set $\{m \mid L_s \vdash m\}$ is compatible with $\prec$ on $\mathsf{hidden}(L_s)$.

For example, in [Backes and Pfitzmann 2004; Janvier et al. 2005] the authors choose $\prec$ to be the order in which the keys are generated: $k \prec k'$ if k has been generated before k'. We denote by $\overline{P}_\prec$ the negation of $P_\prec$. Indeed, an attack in this context is an execution such that the encrypts relation is incompatible with $\prec$.

5.3 Properties that are independent of the notion of key cycle

We show how to decide the existence of key cycles or the conformation to an ordering in polynomial time for solved deducibility constraint systems. Note that the set of messages on which our predicates are applied usually contains all messages sent on the network and possibly some additional intruder knowledge.

We start with statements, that do not depend on which notion of key cycle we choose.

LEMMA 5.6. *Let S be a set of terms, m be a term and k be a key such that $S \vdash m$ and $S \nvdash k$. Then for any plain-text occurrence q of k in m, there is a plain-text occurrence*

q_0 in S such that, if there is key k' with $S \not\vdash k'$, and which protects q_0 in S, then k' protects q in m.

PROOF. We reason by induction on the depth of the proof of $S \vdash m$:

—if the last rule is an axiom, then $m \in S$. We may simply choose $q_0 = q$.

—if the last rule is a decryption, then $S \vdash \mathrm{enc}(m, k'')$ and $S \vdash k''$ for some $k'' \neq k$. Take the position $q_1 = 1 \cdot q$ in $\mathrm{enc}(m, k'')$. It is an occurrence of k. Applying the induction hypothesis we obtain an occurrence q_0 of k in S such that, if there is a key k' with $S \not\vdash k'$ and which protects q_0 in S, then k' protects q_1 in $\mathrm{enc}(m, k'')$. Since $S \not\vdash k'$, it follows that $k'' \neq k'$ and hence k' protects q in m.

—if the last rule is a another rule, we proceed in a similar way as above.

□

As a corollary we obtain the following proposition, which states that, in the passive case, a key cycle can be deduced from a set S only if it already appears in S.

PROPOSITION 5.7. *Let L be a list of ground terms, and $\prec$ a strict partial ordering on a set of keys. The predicate P_{kc} (respectively, P_{skc} or $\overline{P}_{\prec}$) holds on L if and only if L_s contains a key cycle (respectively, L_s contains a strict key cycle, or the encrypts relation on L_s is not compatible with $\prec$).*

PROOF. The right to left direction is trivial since $L_s \subseteq \{m \mid L_s \vdash m\}$.

We will prove the left to right direction only for the key cycle property, the other two properties can be proved in a similar way. Assume that there is no strict partial ordering satisfying the conditions in Definition 5.3 for $\{m \mid L_s \vdash m\}$. In other words, for any strict partial ordering $\prec$ on $\mathsf{hidden}(L_s)$ there is a key k and an occurrence q of k in $\{m \mid L_s \vdash m\}$ such that for any key k', k' protects q in $\{m \mid L_s \vdash m\}$ implies $k' \not\prec k$. Using the previous lemma we can replace $\{m \mid L_s \vdash m\}$ by L_s in the previous sentence, thus obtaining that there is a key cycle in L_s. □

The next lemma will be used to show that $\mathsf{hidden}(L_s\theta)$ does not depend on the solution θ of a solved constraint C.

LEMMA 5.8. *Let $T \Vdash x$ be a constraint of a solved constraint system C, θ a solution of C and m a non-variable term. If $T\theta \vdash m$ then there is a non-variable term u with $\mathcal{V}(u) \subseteq \mathcal{V}(T)$ such that $T \cup \mathcal{V}(T) \vdash u$ and $m = u\theta$.*

PROOF. We write C as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. Consider the index i of the constraint $T \Vdash x$, that is such that $(T_i \Vdash u_i) \in C$, $T_i = T$ and $u_i = x$. The lemma is proved by induction on (i, l) (considering the lexicographical ordering) where l is the length of the proof of $T_i\theta \vdash m$. Consider the last rule of the proof:

—(axiom rule) $m \in T_i\theta$. Then there is $u \in T_i$ such that $m = u\theta$. If u is a variable then there is $j < i$ such that $T_j \Vdash u$ is a constraint of C. We have $T_j\theta \vdash u\theta$. Then by induction hypothesis there is a non-variable term u' with $\mathcal{V}(u') \subseteq \mathcal{V}(T_j)$ such that $T_j \cup \mathcal{V}(T_j) \vdash u'$ and $u\theta = u'\theta$. Hence u' satisfies the conditions.

—(decomposition rule) Suppose the rule is the decryption rule. Then the premises of the rule are $T_i\theta \vdash \mathrm{enc}(m, k)$ and $T_i\theta \vdash k$ for some term k. By induction hypothesis there are non-variable terms u_1 and u_2 with $\mathcal{V}(u_1), \mathcal{V}(u_2) \subseteq \mathcal{V}(T_i)$ such that $T_i \cup \mathcal{V}(T_i) \vdash u_1$,

$T_i \cup \mathcal{V}(T_i) \vdash u_2$, $u_1\theta = \mathrm{enc}(m,k)$ and $u_2\theta = k$. Then $u_1 = \mathrm{enc}(u, u_2')$ with $u\theta = m$ and $u_2'\theta = k$. If u is a variable then, as in the previous case, we find an u' satisfying the conditions. Suppose u is not a variable. We still need to show that $T_i \cup \mathcal{V}(T_i) \vdash u$. If u_2' is a variable then $T_i \cup \mathcal{V}(T_i) \vdash u_2'$ since $u_2' \in \mathcal{V}(T_i)$. If u_2' is not a variable then $u_2'\theta = u_2'$ hence $u_2' = u_2$. In both cases it follows that $T_i \cup \mathcal{V}(T_i) \vdash u$. The projection rule case is simpler and is treated similarly.

—(composition rule) This case follows easily from the induction hypothesis applied on the premises.

□

COROLLARY 5.9. *Let $T \Vdash x$ be a constraint of a solved deducibility constraint system C, and θ, θ' be two solutions of C. Then for any key k, $T\theta \vdash k$ if and only if $T\theta' \vdash k$.*

PROOF. Suppose that $T\theta \vdash k$. From the previous lemma we obtain that there is a non-variable u with $\mathcal{V}(u) \subseteq \mathcal{V}(T)$ such that $T \cup \mathcal{V}(T) \vdash u$ and $k = u\theta$. Since keys are atomic and θ is a ground substitution it follows that $u = k$. Hence $T\theta' \cup \{x\theta' \mid x \in \mathcal{V}(T)\} \vdash k$. So $T\theta' \vdash k$, since θ' is a solution (and thus $T\theta' \vdash x\theta'$ for all $x \in \mathcal{V}(T)$) and by using Lemma 4.5. □

5.4 Decision results

On solved deducibility constraint systems, it is possible to decide in polynomial time, whether an attacker can trigger a key cycle or not, whatever notion of key cycle we consider:

PROPOSITION 5.10. *Let C be a solved deducibility constraint system, L be a list of messages such that $\mathcal{V}(L_s) \subseteq \mathcal{V}(C)$ and $\mathsf{lhs}(C) \subseteq L_s$, and $\prec$ a strict partial ordering on a set of keys. Deciding whether there exists an attack for C and $P(L)$ can be done in $\mathcal{O}(|L|^2)$, for any $P \in \{P_{kc}, P_{skc}, \overline{P}_{\prec}\}$.*

We devote the remaining of this section to the proof of the above proposition.

We know by Proposition 5.7 that it is sufficient to analyze the encrypts (or protects) relation only on $L_s\theta$ (and not on every deducible term), where θ is an arbitrary solution.

We can safely assume that there is exactly one deducibility constraint for each variable. Indeed, eliminating from C all constraints $T' \Vdash x$ for which there is a constraint $T \Vdash x$ in C with $T \subsetneq T'$ we obtain an equivalent deducibility constraint system $C' : \sigma$ is a solution of C' iff it is a solution of C. Let t_x be the term obtained by pairing all terms of T_x (in some arbitrary ordering). We write C as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. We construct the following substitution $\tau = \tau_1 \ldots \tau_n$, and τ_j is defined inductively as follows:

- $\mathrm{dom}(\tau_1) = \{x_1\}$ and $x_1\tau_1 = t_{x_1}$
- $\tau_{i+1} = \tau_i \cup \{{}^{t_{x_{i+1}}\tau_i}/_{x_{i+1}}\}$.

The construction is correct by the definition of deducibility constraint systems. It is clear that τ is a solution of C. We show next that it is sufficient to analyze this particular solution.

Key cycles. We focus first on the property P_{kc}.

LEMMA 5.11. *Let C be a solved deducibility constraint system, L a list of terms such that $\mathcal{V}(L) \subseteq \mathcal{V}(C)$, $\mathsf{lhs}(C) \subseteq L_s$, and assume P is interpreted as P_{kc}. Then there is an attack for C and $P(L)$ if and only if τ is an attack for C and $P(L)$.*

PROOF. We have to prove that if there is no partial ordering satisfying the conditions in Definition 5.3 for the set $L_s\theta$ (according to Proposition 5.7) then there is no partial ordering satisfying the same conditions for $L_s\tau$. Suppose that there is a strict partial ordering $\prec$ which satisfies the conditions for $L_s\tau$. We prove that the same partial ordering does the job for $L_s\theta$.

Let $C' = C \wedge (L_s \Vdash z)$ where z is a new variable. C' is a deducibility constraint system since $\mathsf{lhs}(C) \subseteq L_s$. We write C' as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. We prove by induction on i that for all $k \in \mathsf{hidden}(L_s\theta)$, for all plain-text occurrences q of k in $T_i\theta$ there is a key $k' \in \mathsf{hidden}(L_s\theta)$ such that $k' \prec k$ and k' protects q in $T_i\theta$. It is sufficient to prove this since for $i = n$ we have $T_i = L_s$. Remark also that from Corollary 5.9 applied to $L_s \Vdash z$ we obtain that $\mathsf{hidden}(L_s\theta) = \mathsf{hidden}(L_s\tau)$.

For $i = 1$ we have $T_1 = T_1\theta = T_1\tau$ hence the property is clearly satisfied for θ since it is satisfied for τ.

Let $i > 1$. Consider an occurrence q of a key $k \in \mathsf{hidden}(L_s\theta)$ in a plain-text position of w for some $w \in T_i\theta$. Let $t \in T_i$ such that $w = t\theta$.

If q is a non-variable position in t then it is a position in $t\tau$. And since τ is a solution we have that there is a key $k' \in \mathsf{hidden}(L_s\tau)$ (hence $k' \in \mathsf{hidden}(L_s\theta)$) such that $k' \prec k$ and q is protected by k' in $t\tau$. The key k' cannot occur in some $x\tau$, with $x \in \mathcal{V}(t)$, since otherwise k' is deducible (indeed $x\tau = k'$ since the keys are atomic and $T_x\tau \vdash x\tau$). Hence k' occurs in t. Then k' protects q in t, and thus in w also.

If q is not a non-variable position in t then there is a variable $x_j \in \mathcal{V}(t)$ with $j < i$ such that the occurrence q in $t\theta$ is an occurrence of k in $x_j\theta$ (formally $q = p \cdot q'$ where p is some position of x_j in t and q' is some occurrence of k in $x_j\theta$). Applying Lemma 5.6 we obtain that there is an occurrence q_0 of k in $T_j\theta$ such that if there is a key k' with $T_j\theta \not\vdash k'$ and which protects q_0 in $T_j\theta$ then k' protects q' in $x_j\theta$. The existence of the key k' is assured by the induction hypothesis on $T_j\theta$. Hence k' protects q' in $x_j\theta$ and thus q in w. since otherwise there is $x \in \mathcal{V}(L_s)$ such that $x\tau = k'$, which implies that $k' \notin \mathsf{hidden}(L_s)$. Then q' is a position in $L_s\theta$. Moreover q' protects q in $L_s\theta$.

If q is not a non-variable position in L_s then there is a variable $x \in \mathcal{V}(L_s)$ such that □

Hence we only need to check whether τ is an attack for C and $P(L)$. Let $K = \mathsf{hidden}(L_s\tau)$. We build inductively the sets $K_0 = \emptyset$ and for all $i \geq 1$,

$$K_i = \{k \in K \mid \forall q \in \mathrm{Pos_p}(k, L_s\tau)\, \exists k' \text{ s.t. } k' \text{ protects } q \text{ and } k' \in K_{i-1}\}$$

where $\mathrm{Pos_p}(m, T)$ denotes the plain-text positions of a term m in a set T. Observe that for all $i \geq 0$, $K_i \subseteq K_{i+1}$. This can be proved easily by induction on i. Moreover, since K is finite and $K_i \subseteq K$ for all $i \geq 0$, then there is $l \geq 0$ such that $K_i = K_l$ for all $i > l$.

LEMMA 5.12. *There exists $i \geq 0$ such that $K_i = K$ if and only if $L\tau \in P_{kc}$.*

PROOF. Consider first that there exists $i \geq 0$ such that $K_i = K$. Then take the following strict partial ordering on K: $k' \prec k$ if and only if there is $j \geq 0$ such that $k' \in K_j$ and $k \notin K_j$. Consider a key $k \in K$ and a plain-text occurrence q of k in $L_s\tau$. Then take $l \geq 1$ minimal such that $k \in K_l$. By the definition of K_l there is $k' \in K$ such that k' protects q and $k' \in K_{l-1}$. Since l is minimal $k \notin K_{i-1}$. Hence $k' \prec k$. Thus $L\tau \in P_{kc}$.

Consider now that τ is a solution. Suppose that $K_{i+1} = K_i \subsetneq K$. Let $k \in K \setminus K_{i+1}$. Since $k \notin K_{i+1}$ there is a plain-text occurrence q of k such that for all $k' \in K$ either k' does not protect q, or $k' \notin K_i$. But since τ is a solution, there is $k'' \in K$ such that

k'' protects q and $k'' \prec k$. It follows that $k'' \notin K_i$, and thus $k'' \notin K_{i+1}$. Hence for an arbitrary $k \in K \setminus K_{i+1}$ we have found $k'' \in K \setminus K_{i+1}$ such that $k'' \prec k$. That is, we can build an infinite sequence $\ldots \prec k'' \prec k$ with distinct elements from a finite set – contradiction. So there exists $i \geq 0$ such that $K_i = K$. □

Hence to check whether $L\tau \in P_{kc}$, we only need to construct the sets K_i until $K_{i+1} = K_i$ and then to check whether $K_i = K$. This algorithm is similar to a classical method for finding a topological sorting of vertices (and for finding cycles) of directed graphs. It is also similar to that given by Janvier [Janvier 2006] for the intruder deduction problem considering the deduction system of Laud [Laud 2002].

Regarding the complexity, there are at most $\sharp K$ sets to be build and each set K_i can be constructed in $\mathcal{O}(|L_s\tau|)$. If a DAG-representation of the terms is used then $|L_s\tau| \in \mathcal{O}(|L_s|)$. This gives a complexity of $\mathcal{O}(|K| \times |L_s|)$ for the above algorithm.

Strict key cycles and key orderings.. For the other two properties P_{skc} and $\overline{P}_{\prec}$ we proceed in a similar manner.

LEMMA 5.13. *Let $T \Vdash x$ be a constraint of a solved deducibility constraint system C and θ be a solution. Let m, u, k be terms such that*

$$T\theta \vdash m \text{ and } \mathrm{enc}(u,k) \sqsubseteq m \text{ and } T\theta \nvdash k.$$

Then there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T$ and $v\theta = \mathrm{enc}(u,k)$.

PROOF. We write C as $\bigwedge_i (T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. Consider the index i of the constraint $T \Vdash x$, that is such that $T_i \Vdash u_i \in C$, $T_i = T$ and $u_i = x$. The lemma is proved by induction on (i, l) (lexicographical ordering) where l is the length of the proof of $T_i\theta \vdash m$. Consider the last rule of the proof:

—(axiom rule) $m = t\theta$ for some $t \in T_i$. We can have that either there is $t' \sqsubseteq t$ such that $t'\theta = \mathrm{enc}(u,k)$, or $\mathrm{enc}(u,k) \sqsubseteq y\theta$ for some $y \in \mathcal{V}(t)$. In the first case take $v = t'$, $w = t$. In the second case, by the definition of deducibility constraint systems, there exists $(T_j \Vdash y) \in C$ with $j < i$. Since $T_j\theta \vdash y\theta$ and $T_j\theta \nvdash k$ (since $T_j \subseteq T_i$), we deduce by induction hypothesis that there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T_j$, hence $w \in T_i$ and $v\theta = \mathrm{enc}(u,k)$.

—(decomposition rule) Let m' be the premise of the rule. We have that $T_i\theta \vdash m'$ (with a proof of a strictly smaller length) and $m \sqsubseteq m'$ thus $\mathrm{enc}(u,k) \sqsubseteq m'$. By induction hypothesis, we deduce that there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T_i$ and $v\theta = \mathrm{enc}(u,k)$.

—(composition rule) All cases are similar to the previous one except if $m = \mathrm{enc}(u,k)$ and the rule is $\dfrac{S \vdash x \quad S \vdash y}{S \vdash \mathrm{enc}(x,y)}$. But this case contradicts $T_i\theta \nvdash k$.

□

The following simple lemma is also needed for the proof of Lemma 5.15.

LEMMA 5.14. *Let $T \Vdash x$ be a constraint of a solved deducibility constraint system C, θ be a solution, $k \in \mathsf{hidden}(T\theta)$, and m a term such that $T\theta \vdash m$. If $k\ \rho_1\ m$ then there is $t \in T$ such that $k\ \rho_1\ t$.*

PROOF. We write C as $\bigwedge_i(T_i \Vdash x_i)$, with $1 \leq i \leq n$ and $T_i \subseteq T_{i+1}$. Consider the index i of the constraint $T \Vdash x$, that is such that $(T_i \Vdash u_i) \in C$, $T_i = T$ and $u_i = x$. The lemma is proved by induction on (i, l) (considering the lexicographical ordering) where l is the length of the proof of $T_i\theta \vdash m$. Consider the last rule of the proof:

—(axiom rule) $m \in T_i\theta$ or m a public constant. If m is a public constant then $k \neq m$ since $k \in \mathsf{hidden}(T\theta)$. Thus there is $t \in T_i$ such that $m = t\theta$. If $k \,\rho_1\, t$ then we're done. Otherwise there is a variable $y \in \mathcal{V}(t)$ such that $k \,\rho_1\, y\theta$. Also, there is $j < i$ such that $T_j \Vdash y$ is a constraint of C. Then, by induction hypothesis, there is $t' \in T_j$, hence in T_i, such that $k \,\rho_1\, t'$.

—(composition or decomposition rule) By inspection of all the composition and decomposition rules we observe that there is always a premise $T_i\theta \vdash m'$ with $k \,\rho_1\, m'$ for some term m'. The conclusion follows then directly from the induction hypothesis.

□

The following lemma shows that it is sufficient to analyze τ when checking the properties P_{skc} and $\overline{P}_{\prec}$.

LEMMA 5.15. *Let C be a solved deducibility constraint system, L a list of terms such that $\mathcal{V}(L) \subseteq \mathcal{V}(C)$ and $\mathsf{lhs}(C) \subseteq L_s$, and θ a solution of C. For any $k, k' \in \mathsf{hidden}(L_s\theta)$, if k encrypts k' in $L_s\theta$ then k encrypts k' in $L_s\tau$.*

PROOF. Remember that $\mathsf{hidden}(L_s\theta) = \mathsf{hidden}(L_s\tau)$ (Corollary 5.9).

Consider two keys $k, k' \in \mathsf{hidden}(L_s\theta)$ such that k encrypts k' in $L_s\theta$. Then there are terms u, u' such that $u' \in L_s\theta$, $\mathrm{enc}(u, k) \sqsubseteq u'$ and $k' \,\rho_1\, u$. We can have that either (first case) there are v, w such that $v \sqsubseteq w \in L_s$, v non-variable and $\mathrm{enc}(u, k) = v\theta$, or (second case) $\mathrm{enc}(u, k) \sqsubseteq x\theta$ with $x \in \mathcal{V}(L_s)$. In the second case, consider the constraint $(T_x \Vdash x) \in C$. We have $T_x\theta \vdash x\theta$. Hence we can apply Lemma 5.13 for $x\theta$, u and k to obtain that there exists a non-variable term v such that $v \sqsubseteq w$ for some $w \in T_x$ and $v\theta = \mathrm{enc}(u, k)$. Hence, in both cases, we obtained that there is a non-variable term $v \in St(L_s)$ (since $T_x \subseteq L_s$) such that $v\theta = \mathrm{enc}(u, k)$. Thus there is v_0 such that $v = \mathrm{enc}(v_0, k)$. Indeed, otherwise $v = \mathrm{enc}(v_0, y)$ for some $y \in \mathcal{V}(L_s)$, hence $y \in \mathcal{V}(C)$. Since C is solved we have $T_y\sigma \vdash y\sigma$. But $y\sigma = k$, contradicting $k \in \mathsf{hidden}(L_s\theta)$.

We have $v_0\theta = u$. Since $k' \,\rho_1\, u$ and k' is a name or a variable, we can have that $k' \,\rho_1\, v_0$, or $k' \,\rho_1\, y\theta$ for some $y \in \mathcal{V}(v_0)$. If $k' \,\rho_1\, v_0$ then k encrypts k' in L_s, hence in $L_s\tau$ also. If $k' \,\rho_1\, y\theta$ then from the previous lemma $k' \,\rho_1\, t$ for some $t \in T_y$, and hence $k' \,\rho_1\, y\tau$. Therefore in both cases we have that k encrypts k' in $L_s\tau$. □

We deduce that deciding whether there is an attack for C and $P(L)$, when P is interpreted as P_{skc},can be done simply by deciding whether the restriction of the relation $\rho_e^{L_s\tau}$ to $K \times K$ is cyclic.

Deciding whether there is an attack for C and $P(L)$, when P is interpreted as $\overline{P}_{\prec}$, can be done by deciding whether the restriction to $K \times K$ of the relation $\rho_e^{L_s\tau}$ has the following property Q: there are $k, k' \in K$ such that $k\rho_e^{L_s\tau}k'$ and $k \preceq k'$.

Checking the cyclicity of the relation $\rho_e^{L_s\tau}$ reduces to checking the cyclicity of the corresponding directed graph, using a classic algorithm in $\mathcal{O}(|K|^2)$. Then, checking the property Q can be performed by analyzing all pairs $(k, k') \in K \times K$ hence also in $\mathcal{O}(|K|^2)$.

Verifying any of the three properties requires a preliminary step of computing $K = \mathsf{hidden}(L_s\tau)$. Computing deducible subterms can be performed in linear time, hence this

computation step requires $\mathcal{O}(|L_s\tau|)$. $|L_s\tau| \leq |L_s|+|\tau| \leq |L_s|+\mathcal{O}(|C|)$. If $\mathsf{lhs}(C) \subseteq L_s$, then $|L_s\tau| = \mathcal{O}(|L|)$. It follows that the complexity of deciding whether there is an attack for C and $P(L)$ is $\mathcal{O}(|L|^2)$, when P is interpreted as P_{kc}, P_{skc} or $\overline{P}_{\prec}$.

5.5 NP-completeness

Let C be a deducibility constraint system and L a list of terms such that $\mathcal{V}(L_s) \subseteq \mathcal{V}(C)$ and $\mathsf{lhs}(C) \subseteq L_s$. The NP membership of deciding whether there is an attack for C and $P(L)$ (for our 3 possible interpretations of P) follows immediately from Corollary 4.18 and Proposition 5.10.

NP-hardness is obtained by adapting the construction for NP-hardness provided in [Rusinowitch and Turuani 2003]. More precisely, we consider the reduction of the 3SAT problem to our problem. For any 3SAT Boolean formula we construct a protocol such that the intruder can deduce a key cycle if and only if the formula is satisfiable. The construction is the same as in [Rusinowitch and Turuani 2003] (pages 15 and 16) except that, in the last rule, the participant responds with the term $\mathrm{enc}(k,k)$, for some fresh key k (initially secret), instead of $Secret$. Then it is easy to see that the only way to produce a key cycle on a secret key is to play this last rule which is equivalent, using [Rusinowitch and Turuani 2003], to the satisfiability of the corresponding 3SAT formula.

6. AUTHENTICATION-LIKE PROPERTIES

We propose a simple decidable logic for security properties. This logic enables in particular to specify authentication-like properties.

6.1 A simple logic

The logic enables terms comparisons and is closed under Boolean connectives.

Definition 6.1. The logic $\mathcal{L}$ is inductively defined by:

$$\phi ::= [m_1 = m_2] \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \bot \qquad m_1, m_2 \text{ terms}$$

$\mathcal{V}(\phi)$ is the set of variables occurring in its atomic formulas.

$\sigma \models [m_1 = m_2]$ if $m_1\sigma$ and $m_2\sigma$ are identical terms. $\sigma \not\models \bot$. This satisfaction relation is extended to any of the above formulas, interpreting the Boolean connectives as usual.

Example 6.2. Let us consider again the authentication property introduced in Example 3.8. There is an attack on authentication between A and B if A and B do not agree on the nonce n'_a sent by A for B, that is if $x = n'_a$ at the end of the run of the protocol. This can be expressed by the following formula

$$\phi_1 = [x \neq n'_a]$$

The substitution σ_1 (assigning x to n_a) is an attack for C'_1 (defined in Example 3.8) and ϕ_1 and demonstrates a failure of authentication.

More sophisticated properties can be expressed using the logic $\mathcal{L}$. For example, when two sessions of the same role are executed, one can expressed that an agent has received *exactly once* the right nonce n_a, with the following formula.

$$\phi_2 = ([x_1 = n_a] \wedge [x_2 \neq n_a]) \vee ([x_1 \neq n_a] \wedge [x_2 = n_a])$$

where x_1 (resp. x_2) represents the nonce received by the agent in the first (resp. second) session.

We can also express properties of the form: if two agents agree on some term u, they also agree on some term v. This can be indeed modeled by the formula

$$\phi_3 = [u_1 = u_2] \rightarrow [v_1 = v_2]$$

where u_1 (resp. u_2) represents the view of u by the first (resp. second) agent and v_1 (resp. v_2) represents the view of v by the first (resp. second) agent. The formula $A \rightarrow B$ is the usual notation for the formula $\neg A \vee B$.

6.2 Decidability

THEOREM 6.3. *Let C be a deducibility constraint system and ϕ be a formula of $\mathcal{L}$. Deciding whether there is an attack for C and ϕ can be performed in non-deterministic polynomial time.*

PROOF. First, choosing non-deterministically ϕ_1 or ϕ_2 in any subformula $\phi_1 \vee \phi_2$, we may, w.l.o.g. only consider the case where ϕ is a conjunction $\bigwedge_j [u_j = u'_j] \wedge \phi_d$, where $\phi_d = \bigwedge_l [v_l \neq v'_l]$.

Let σ be a mgu (idempotent, which does not introduce new variables) of $\bigwedge_j u_j = u'_j$. The deducibility constraint system C has a joined solution with ϕ if and only if $C\sigma$ and $\phi_d\sigma$ have a common solution. As in the previous sections, we choose a representation of expressions, such that applying a mgu of subterms of an expression e on e does not increase the size of the expression e.

We are now left to the case where we have to decide whether a deducibility constraint system has a solution together with a property of the form $\phi = \bigwedge_{i=1}^{k} [u_i \neq v_i]$.

Applying Theorem 4.3, there exists a solution θ of C and ϕ if and only if there exist a deducibility constraint system C' in solved form and substitutions σ, θ' such that $\theta = \sigma\theta'$, $C \rightsquigarrow_\sigma^* C'$ and θ' is an attack for C' and $\phi\sigma$. Thus, we are now left to decide whether there exists a solution to a solved constraint system C' and a formula $\phi\sigma$ of the form $\phi\sigma = \bigwedge_{i=1}^{k} [u_i \neq v_i]$.

If, for some i, u_i is identical to v_i, then there is clearly no solution. We claim that, otherwise, there is always a solution. This is an independence of disequation lemma (as in [Colmerauer 1984] for instance), and the proof is similar to other independence of disequations lemmas:

LEMMA 6.4. *Let C be a solved deducibility constraint system and ϕ be the formula $t_1 \neq u_1 \wedge \ldots \wedge t_n \neq u_n$ such that $\mathcal{V}(\phi) \subseteq \mathcal{V}(C)$ and, for every i, t_i is not identical to u_i. Then there is always a solution θ of C and ϕ.*

This is proved by induction on the number of variables of ϕ. In the base case, there is no variable and the result is trivial as ϕ is a tautology.

Let T_0 be the smallest left-hand side of C. T_0 must be a non empty set of ground terms. Note that there is an infinite set of deducible terms from T_0.

Let $x \in \mathcal{V}(\phi)$. For each i, either $t_i = u_i$ has no solution, in which case $t_i \neq u_i$ is always satisfied, or else let $S = \{x\sigma_i \mid \sigma_i = \mathrm{mgu}(t_i, u_i)\}$. We choose t_x such that $T \vdash t_x$ and $t_x \notin S$. This is possible since S is finite and there are infinitely many terms deducible from T. Now, for every i, $t_i[{}^{t_x}/_x]$ is not identical to $u_i[{}^{t_x}/_x]$ by construction. Hence, we may apply the induction hypothesis to $\phi[{}^{t_x}/_x]$ and conclude. □

7. TIMESTAMPS

For modeling timestamps, we introduce a new sort $\mathsf{Time} \subseteq \mathsf{Msg}$ for time and we assume an infinite number of names of sort Time, represented by rational numbers or integers. We assume that the only two sorts are Time and Msg. Any value of time should be known to an intruder, that is why we add to the deduction system the rule $\frac{}{S \vdash a}$ for any name a of sort Time. All the previous results can be easily extended to such a deduction system since ground deducibility remains decidable in polynomial time.

To express relations between timestamps, we use timed constraints.

Definition 7.1. An *integer timed constraint* or a *rational timed constraint* T is a conjunction of formulas of the form

$$\Sigma_{i=1}^{k} \alpha_i x_i \ltimes \beta,$$

where the α_i and β are rational numbers, $\ltimes \in \{<, \leq\}$, and the x_i are variables of sort Time. A *solution* of a rational (resp. integer) timed constraint T is a closed substitution $\sigma = \{^{c_1}/_{x_1}, \ldots, ^{c_k}/_{x_k}\}$, where the c_i are rationals (resp. integers), that satisfies the constraint.

Such timed properties can be used for example to say that a timestamp x_1 must be fresher than a timestamp x_2 ($x_1 \geq x_2$) or that x_1 must be at least 30 seconds fresher than x_2 ($x_1 \geq x_2 + 30$).

Example 7.2. We consider the Wide Mouthed Frog Protocol [Clark and Jacob 1997].

$$\begin{array}{l} A \to S : \ A, \mathrm{enc}(\langle T_a, B, K_{ab}\rangle, K_{as}) \\ S \to B : \ \mathrm{enc}(\langle T_s, A, K_{ab}\rangle, K_{bs}) \end{array}$$

A sends to a server S a fresh key K_{ab} intended for B. If the timestamp T_a is fresh enough, the server answers by forwarding the key to B, adding its own timestamps. B simply checks whether this timestamp is older than any other message he has received from S. As explained in [Clark and Jacob 1997], this protocol is flawed because an attacker can use the server to keep a session alive as long as he wants by replaying the answers of the server.

This protocol can be modeled by the following deducibility constraint system:

$$S_1 \stackrel{\mathsf{def}}{=} \{a, b, s, \langle a, \mathrm{enc}(\langle 0, b, k_{ab}\rangle, k_{as})\rangle\} \ \Vdash \ \langle a, \mathrm{enc}(\langle x_{t_1}, b, y_1\rangle, k_{as})\rangle, x_{t_2} \quad (6)$$

$$S_2 \stackrel{\mathsf{def}}{=} S_1 \cup \{\mathrm{enc}(\langle x_{t_2}, a, y_1\rangle, k_{bs})\} \ \Vdash \ \langle b, \mathrm{enc}(\langle x_{t_3}, a, y_2\rangle, k_{bs})\rangle, x_{t_4} \quad (7)$$

$$S_3 \stackrel{\mathsf{def}}{=} S_2 \cup \{\mathrm{enc}(\langle x_{t_4}, b, y_2\rangle, k_{as})\} \ \Vdash \ \langle a, \mathrm{enc}(\langle x_{t_5}, b, y_3\rangle, k_{as})\rangle, x_{t_6} \quad (8)$$

$$S_4 \stackrel{\mathsf{def}}{=} S_3 \cup \{\mathrm{enc}(\langle x_{t_6}, a, y_3\rangle, k_{bs})\} \ \Vdash \ \mathrm{enc}(\langle x_{t_7}, a, k_{ab}\rangle, k_{bs}) \quad (9)$$

where y_1, y_2, y_3 are variables of sort Msg and $x_{t_1}, \ldots, x_{t_7}$ are variables of sort Time. We add explicitly the timestamps emitted by the agents on the right hand side of the constraints (that is in the messages expected by the participants) since the intruder can schedule the message transmission whenever he wants. Note that on the right hand side of constraints we do have terms, but by abuse of notation we have omitted the pairing function symbol.

Initially, the intruder simply knows the names of the agents and A's message at time 0. Then S answers alternatively to requests from A and B. Since the intruder controls the network, the messages can be scheduled as slow (or fast) as the intruder needs it. The server S should not answer if A's timestamp is too old (let's say older than 30 seconds)

thus S's timestamp cannot be too much delayed (no more than 30 seconds). This means that we should have $x_{t_2} \leq x_{t_1} + 30$. Similarly, we should have $x_{t_4} \leq x_{t_3} + 30$ and $x_{t_6} \leq x_{t_5} + 30$. The last rule corresponds to B's reception. In this scenario, B does not perform any check on the timestamp since it is the first message he receives.

We say that there is an attack if there is a joined solution of the deducibility constraint system and the previously mentioned time constraints together with $x_{t_7} \geq 30$. This last constraint expresses that the timestamp received by B is too large to come from A. Altogether, the time constraint becomes $x_{t_2} \leq x_{t_1} + 30 \ \wedge \ x_{t_4} \leq x_{t_3} + 30 \ \wedge \ x_{t_6} \leq x_{t_5} + 30 \wedge \ x_{t_7} \geq 30$. Then the substitution corresponding to the attack is

$$\sigma = \{{}^{k_{ab}}\!/_{y_1}, {}^{k_{ab}}\!/_{y_2}, {}^{k_{ab}}\!/_{y_3}, {}^{k_{ab}}\!/_{y_4}, {}^{0}\!/_{x_{t_1}}, {}^{30}\!/_{x_{t_2}}, {}^{30}\!/_{x_{t_3}}, {}^{60}\!/_{x_{t_4}}, {}^{60}\!/_{x_{t_5}}, {}^{90}\!/_{x_{t_6}}, {}^{90}\!/_{x_{t_7}}\}.$$

PROPOSITION 7.3. *There is an attack to a solved deducibility constraint system and a time constraint T iff T has a solution.*

PROOF SKETCH. Let C be a solved deducibility constraint system, and T a timed constraint. Let $y_1, \ldots, y_n$ be the variables of sort Msg in C and $x_1, \ldots, x_k$ the variables of sort Time in C. Clearly, any substitution σ of the form $y_i\sigma = u_i$ where $u_i \in S_i$ for some $(S_i \Vdash y_i) \in C$ and $x_i\sigma = t_i$ for t_i any constant of sort Time is a solution of C. Let σ' be the restriction of σ to the timed variables $x_1, \ldots, x_k$.

σ is an attack for C and T if and only if σ' is a solution to T. Thus there exists an attack for C and T if and only if T is satisfiable. □

COROLLARY 7.4. *Deciding whether a deducibility constraint system, together with a time constraint, has a solution is NP-complete.*

PROOF. The NP membership follows from the NP membership of time constraint satisfiability, Theorem 4.3 and Proposition 7.3.

NP-hardness directly follows from the NP-hardness of deducibility constraint system solving, considering an empty timed constraint. □

8. CONCLUSIONS

We have shown how, revisiting the approach of [Comon-Lundh and Shmatikov 2003; Rusinowitch and Turuani 2003], we can preserve the set of solutions, instead of only deciding the satisfiability. We also derived NP-completeness results for some security properties: key-cycles, authentication, time constraints.

Since the constraint-based approach [Comon-Lundh and Shmatikov 2003; Rusinowitch and Turuani 2003] has already been implemented in AVISPA [Armando et al. 2005], it is likely that we can, with only slight efforts, adapt this implementation to the case of key cycles and timestamps.

More generally, we would like to take advantage of our result to derive decision procedures for even more security properties. A typical example would be the combinations of several properties. Also, we could investigate non-trace properties such as anonymity or guessing attacks, for which there are very few decision results (only [Baudet 2005], whose procedure is quite complex).

Regarding key cycles, our approach is valid for a bounded number of sessions only. Secrecy is undecidable in general [Durgin et al. 2004] for an unbounded number of sessions. Such an undecidability result could be easily adapted to the problem of detecting key cycles. Secrecy is decidable for several classes of protocols [Ramanujam and Suresh 2003;

Comon-Lundh and Cortier 2003; Blanchet and Podelski 2003; Verma et al. 2005] and an unbounded number of sessions. We plan to investigate how such fragments could be used to decide key cycles.

Acknowledgments.. We are particularly grateful to Michael Backes, Michaël Rusinowitch, Stéphanie Delaune, and Bogdan Warinschi for their very helpful suggestions.

REFERENCES

ABADI, M. AND ROGAWAY, P. 2002. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology 2*, 103–127.

ADÃO, P., BANA, G., HERZOG, J., AND SCEDROV, A. 2005. Soundness of formal encryption in the presence of key-cycles. In *Proc. of the 10th European Symposium on Research in Computer Security (ESORICS'05)*. Lecture Notes in Computer Science, vol. 3679. Springer Verlag, 374–396.

AMADIO, R. AND LUGIEZ, D. 2000. On the reachability problem in cryptographic protocols. In *Proc. of the 11th Int. Conf. on Concurrency Theory (CONCUR'00)*. Lecture Notes in Computer Science, vol. 1877. Springer Verlag, 380–394.

ARMANDO, A., BASIN, D., BOICHUT, Y., CHEVALIER, Y., COMPAGNA, L., CUELLAR, J., DRIELSMA, P. H., HÉAM, P., KOUCHNARENKO, O., MANTOVANI, J., MÖDERSHEIM, S., VON OHEIMB, D., RUSINOWITCH, M., SANTIAGO, J., TURUANI, M., VIGANÒ, L., AND VIGNERON, L. 2005. The AVISPA tool for the automated validation of internet security protocols and applications. In *Proc. of the Computer Aided Verification (CAV'05)*. Lecture Notes in Computer Science, vol. 3576. Springer Verlag.

BACKES, M. AND PFITZMANN, B. 2004. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*. IEEE Computer Society Press, 204–218.

BACKES, M., PFITZMANN, B., AND SCEDROV, A. 2007. Key-dependent message security under active attacks – BRSIM/UC-soundness of symbolic encryption with key cycles. In *Proc. of the 20th IEEE Computer Security Foundations Symposium (CSF'07)*. IEEE Computer Society Press. Preprint on IACR ePrint 2005/421.

BAUDET, M. 2005. Deciding security of protocols against off-line guessing attacks. In *Proc. of the 12th ACM Conf. on Computer and Communication Security (CCS'05)*. ACM Press, 16–25.

BELLARE, M. AND ROGAWAY, P. 1993. Entity authentication and key distribution. In *Proc. of the 13th Annual Int. Conf. on Advances in Cryptology (CRYPTO'93)*. Lecture Notes in Computer Science, vol. 773. Springer Verlag, 232–249.

BLANCHET, B. 2001. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. of the 14th IEEE Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, 82–96.

BLANCHET, B. AND PODELSKI, A. 2003. Verification of cryptographic protocols: Tagging enforces termination. In *Foundations of Software Science and Computation Structures (FoSSaCS'03)*, A. Gordon, Ed. Lecture Notes in Computer Science, vol. 2620. Springer Verlag, 136–152.

BOZGA, L., ENE, C., AND LAKHNECH, Y. 2004. A symbolic decision procedure for cryptographic protocols with time stamps. In *Proc. of the 15th Int. Conf. on Concurrency Theory (CONCUR'04)*. Lecture Notes in Computer Science, vol. 3170. Springer Verlag, 177–192.

BURSUC, S., COMON-LUNDH, H., AND DELAUNE, S. 2007. Associative-commutative deducibility constraints. In *Proc. of the 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS'07)*. Lecture Notes in Computer Science, vol. 4393. Springer Verlag, 634–645.

CLARK, J. AND JACOB, J. 1997. A survey of authentication protocol literature. Available at `http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps`.

COLMERAUER, A. 1984. Equations and inequations on finite and infinite trees. In *Proc. of the Int. Conf. on Fifth Generation Computer Systems (FGCS'84)*. 85–99.

COMON-LUNDH, H. AND CORTIER, V. 2003. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'03)*. Lecture Notes in Computer Science, vol. 2706. Springer Verlag, 148–164.

COMON-LUNDH, H. AND SHMATIKOV, V. 2003. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proc. of the 18th Annual IEEE Symposium on Logic in Computer Science (LICS'03)*. IEEE Computer Society Press, 271–280.

CORIN, R. 2006. Analysis models for security protocols. Ph.D. thesis, University of Twente, The Netherlands.

CORIN, R. AND ETALLE, S. 2002. An improved constraint-based system for the verification of security protocols. In *Proc. of the 9th Int. Symposium on Static Analysis (SAS'02)*. Lecture Notes in Computer Science, vol. 2477. Springer Verlag, 326–341.

CORIN, R. J., SAPTAWIJAYA, A., AND ETALLE, S. 2005. PS-LTL for constraint-based security protocol analysis. In *Proc. of the 21st Int. Conf. on (ICLP'05)*. Lecture Notes in Computer Science, vol. 3668. Springer Verlag, 439–440.

CORTIER, V., DELAITRE, J., AND DELAUNE, S. 2007. Safely composing security protocols. In *Proc. of the 27th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*. Lecture Notes in Computer Science, vol. 4855. Springer Verlag, 352–363.

CORTIER, V., KREMER, S., KÜSTERS, R., AND WARINSCHI, B. 2006. Computationally sound symbolic secrecy in the presence of hash functions. In *Proc. of the 26th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*. Lecture Notes in Computer Science, vol. 4337. Springer Verlag, 176–187.

CORTIER, V. AND ZĂLINESCU, E. 2006. Deciding key cycles for security protocols. In *Proc. of the 13th Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*. Lecture Notes in Artificial Intelligence, vol. 4246. Springer Verlag, 317–331.

CREMERS, C. 2008. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *Proc. of the 20th Int. Conf. Computer Aided Verification (CAV'08)*. Lecture Notes in Computer Science, vol. 5123. Springer Verlag, 414–418.

DURGIN, N., LINCOLN, P., AND MITCHELL, J. 2004. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security 12*, 2, 247–311.

DURGIN, N., LINCOLN, P., MITCHELL, J., AND SCEDROV, A. 1999. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*.

GOLDWASSER, S. AND MICALI, S. 1984. Probabilistic encryption. *Journal of Computer and System Sciences 28*, 270–299.

HOFHEINZ, D. AND UNRUH, D. 2008. Towards key-dependent message security in the standard model. In *EUROCRYPT 2008*. Lecture Notes in Computer Science, vol. 4965. Springer Verlag, 108–126. Preprint on IACR ePrint 2007/333.

JANVIER, R. 2006. Lien entre modèles symboliques et computationnels pour le protocoles cryptographiques utilisant des hachage. Ph.D. thesis, Université Joseph Fourier, Grenoble.

JANVIER, R., LAKHNECH, Y., AND MAZARE, L. 2005. (De)Compositions of Cryptographic Schemes and their Applications to Protocols. Cryptology ePrint Archive, Report 2005/020.

LAUD, P. 2002. Encryption cycles and two views of cryptography. In *Proc. of the Nordic Workshop on Secure IT Systems (NORDSEC'02)*.

LOWE, G. 1996. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. of the 2nd Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*. Lecture Notes in Computer Science, vol. 1055. Springer Verlag, 147–166.

LOWE, G. 1998. Towards a completeness result for model checking of security protocols. In *Proc. of the 11th IEEE Computer Security Foundations Workshop (CSFW'98)*. IEEE Computer Society Press.

MICCIANCIO, D. AND WARINSCHI, B. 2004a. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security 12*, 1, 99–129. Preliminary version in WITS'02.

MICCIANCIO, D. AND WARINSCHI, B. 2004b. Soundness of formal encryption in the presence of active adversaries. In *Proc. of the 1st Theory of Cryptography Conference (TCC'04)*. Lecture Notes in Computer Science, vol. 2951. Springer Verlag, 133–151.

MILLEN, J. AND SHMATIKOV, V. 2001. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of the 8th ACM Conf. on Computer and Communication Security (CCS'01)*. ACM Press, 166–175.

NEEDHAM, R. M. AND SCHROEDER, M. D. 1978. Using encryption for authentication in large networks of computers. *Communications of the ACM 21*, 12, 993–999.

RAMANUJAM, R. AND SURESH, S. P. 2003. Tagging makes secrecy decidable for unbounded nonces as well. In *Proc. of the 23rd Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'03)*. Lecture Notes in Computer Science, vol. 2914. Springer Verlag, 363–374.

RAMANUJAM, R. AND SURESH, S. P. 2005. Decidability of context-explicit security protocols. *Journal of Computer Security 13*, 1, 135–165.

RUSINOWITCH, M. AND TURUANI, M. 2001. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th IEEE Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, 174–190.

RUSINOWITCH, M. AND TURUANI, M. 2003. Protocol insecurity with finite number of sessions and composed keys is NP-complete. *Theoretical Computer Science 299*, 451–475.

SYVERSON, P. AND MEADOWS, C. 1996. A formal language for cryptographic protocol requirements. *Designes, Codes and Cryptography 7*, 1-2, 27–59.

VERMA, K. N., SEIDL, H., AND SCHWENTICK, T. 2005. On the complexity of equational Horn clauses. In *Proc. of the 22th Int. Conf. on Automated Deduction (CADE'05)*. Lecture Notes in Computer Science. Springer Verlag, 337–352.

Received August 2007; accepted April 2008